Configurer et Paramètrer GRBL

C’est une question qui revient souvent, après avoir installé GRBL. Comment dois-je faire pour configurer et paramètrer GRBL pour ma machine? Bien sûr, chaque machine est différente, dans ses courses, dimensions, moteurs, vitesse dont elle est capable, etc.…

Il y a deux types différents de “réglages” possible: “En dur”, à la compilation, en modifiant des valeurs dans un fichier, et en interactif, alors que le programme s’exécute. Dans la suite de cet article, et pour mieux différencier les deux, on va appeler les réglages en dur “configuration”, et ceux en interactif “paramétrage”, même si cette différence de vocabulaire est un peu arbitraire. J’espère simplement que cela contribuera à la compréhension.

Configuration

On va commencer par décrire ce qui ne l’est pas très souvent, la modification de valeurs dans certains fichiers qui seront compilés avec GRBL lorsque vous le flashez sur l’Arduino. Si vous avez déjà installé GRBL dans votre Arduino, vous l’avez certainement compilé et flashé tel que, sans faire de modifications. Très bien, cela fonctionne, et ce n’est pas un problème particulier. Il peut cependant être utile de savoir comment modifier la configuration de base, et en quoi cela peut vous aider.

Dans votre IDE (Visual Studio pour moi), ouvrez le fichier config.h qui doit se trouver dans votre répertoire GRBL. C’est un assez long fichier (+- 400 lignes) extrêmement bien commenté, en Anglais bien sûr. Nous allons isoler ici les paramètres de configuration les plus importants pour s’adapter au mieux à votre machine spécifique.

Tout d’abord, si vous liez en parallèle 2 interrupteurs de butée pour 1 même axe, au lieu d’avoir un switch câblé séparément pour chaque extrémité, il faut dé-commenter la ligne commençant par define, ci-dessous, en supprimant les “//”

```
// If your machine has two limits switches wired in parallel to one ax
is, you will need to enable  // this feature. Since the two switches a
re sharing a single pin, there is no way for Grbl to tell  // which on
e is enabled. This option only effects homing, where if a limit is eng
aged, Grbl will  // alarm out and force the user to manually disengage
 the limit switch. Otherwise, if you have one  // limit switch for eac
h axis, don't enable this option. By keeping it disabled, you can perf
orm a  // homing cycle while on the limit switch and not have to move
the machine off of it.  // #define LIMITS_TWO_SWITCHES_ON_AXES
```

Ensuite, et toujours en liaison avec vos interrupteurs de fin de course, il faut définir si vous câblez vos switches en “normalement ouvert”, où en “normalement fermé”. Par défaut, GRBL est défini pour utiliser des switches normalement ouverts. Dé-commentez la ligne pour un

fonctionnement en normalement fermé:

```
// Inverts pin logic of the control command pins. This essentially mea
ns when this option is enabled  // you can use normally-closed switche
s, rather than the default normally-open switches.  // NOTE: If you re
quire individual control pins inverted, keep this macro disabled and s
imply alter  // the CONTROL_INVERT_MASK definition in cpu_map.h files.
  // #define INVERT_ALL_CONTROL_PINS // Default disabled. Uncomment to
 enable.
```

Depuis peu, GRBL est capable de gérer (s'il reçoit les commandes G-Code correspondantes) la vitesse de rotation de la broche. Si vous vous servez de cette fonction, laissez là activée (elle l'est par défaut), sinon désactivez-là. Si vous la laissez activée, il vous faudra aussi définir la plage de vitesses dont votre broche est capable:

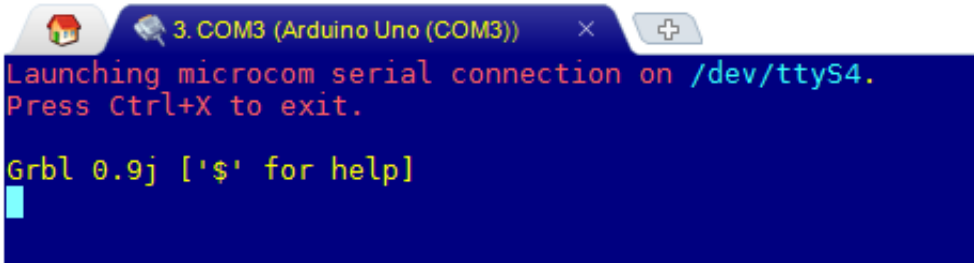
```
// Enables variable spindle output voltage for different RPM values. O
n the Arduino Uno, the spindle  // enable pin will output 5V for maxim
um RPM with 256 intermediate levels and 0V when disabled.  // NOTE: IM
PORTANT for Arduino Unos! When enabled, the Z-limit pin D11 and spindl
e enable pin D12 switch!  // The hardware PWM output on pin D11 is req
uired for variable spindle output voltages.  #define VARIABLE_SPINDLE
// Default enabled. Comment to disable.    // Used by the variable spi
ndle output only. These parameters set the maximum and minimum spindle
 speed  // "S" g-code values to correspond to the maximum and minimum
pin voltages. There are 256 discrete and  // equally divided voltage b
ins between the maximum and minimum spindle speeds. So for a 5V pin, 1
000  // max rpm, and 250 min rpm, the spindle output voltage would be
set for the following "S" commands:  // "S1000" @ 5V, "S250" @ 0.02V,
and "S625" @ 2.5V (mid-range). The pin outputs 0V when disabled.  // B
G Note: As the spindle max speed is said to be 12K, that's what I set
here. Should never go that fast in the GCode, but I will test.    #def
ine SPINDLE_MAX_RPM 18000.0 // Max spindle RPM. This value is equal to
 100% duty cycle on the PWM.  #define SPINDLE_MIN_RPM 0.0 // Min spind
le RPM. This value is equal to (1/256) duty cycle on the PWM
```

Jusqu'à présent, je n'ai pas eu besoin de m'intéresser de près aux autres paramètres du fichier config.h, mais sachez que vous avez peut-être là la solution à certaines difficultés que vous pourriez rencontrer en utilisant GRBL

Paramétrage

Le paramétrage se fait alors que l'Arduino est branché et que vous êtes en liaison série avec

lui. Il existe beaucoup de logiciels différents permettant d'envoyer le G-Code et de contrôler GRBL, que ce soit depuis un PC, un Mac où une machine Linux. Pour éviter d'entrer de suite dans d'autres considérations de choix pour ces logiciels, nous allons paramétrer GRBL le plus simplement du monde avec un terminal série, en envoyant manuellement les commandes.



Lorsque vous vous connectez avec un terminal série sur votre Arduino, si le flashage s'est passé correctement, vous devriez obtenir une réponse `Grbl 0.9j ['$' for help]`, tel que vous pouvez le voir ici.

Pour lister l'ensemble des paramètres qui vous permettront d'adapter Grbl aux spécificités de votre machine, il vous suffit de taper $$. Grbl va répondre en renvoyant la liste visible ci-dessous.

Paramètres 100, 101 et 102, nombre de pas par millimètre.

Nous allons tout d'abord nous intéresser aux paramètres 100, 101 et 102. Ces paramètres,

comme indiqué dans le commentaire

```
Grbl 0.9j ['$' for help]
$0=10 (step pulse, usec)
$1=25 (step idle delay, msec)
$2=0 (step port invert mask:00000000)
$3=0 (dir port invert mask:00000000)
$4=0 (step enable invert, bool)
$5=0 (limit pins invert, bool)
$6=0 (probe pin invert, bool)
$10=3 (status report mask:00000011)
$11=0.020 (junction deviation, mm)
$12=0.002 (arc tolerance, mm)
$13=0 (report inches, bool)
$20=0 (soft limits, bool)
$21=0 (hard limits, bool)
$22=0 (homing cycle, bool)
$23=0 (homing dir invert mask:00000000)
$24=25.000 (homing feed, mm/min)
$25=500.000 (homing seek, mm/min)
$26=250 (homing debounce, msec)
$27=1.000 (homing pull-off, mm)
$100=250.000 (x, step/mm)
$101=250.000 (y, step/mm)
$102=250.000 (z, step/mm)
$110=600.000 (x max rate, mm/min)
$111=600.000 (y max rate, mm/min)
$112=600.000 (z max rate, mm/min)
$120=10.000 (x accel, mm/sec^2)
$121=10.000 (y accel, mm/sec^2)
$122=10.000 (z accel, mm/sec^2)
$130=550.000 (x max travel, mm)
$131=350.000 (y max travel, mm)
$132=130.000 (z max travel, mm)
ok
```

entre parenthèses, définissent le nombre de pas (des moteurs pas à pas) nécessaires pour parcourir une distance de 1 millimètre sur l'axe indiqué. La plupart du temps, 100 et 101 (X et Y) seront identiques, alors que Z (102) aura une valeur différente. Pour vous faciliter la tache, vous trouverez au bas de l'article un calculateur pour trouver cette valeur, mais la formule est assez simple:

Valeur = Nombre de pas (du moteur) pour faire 360°/Pas de la Visse sans fin (où nombre de dents de la poulie * espace dents chaîne où courroie)

Résultat = Valeur x Micro Pas.

Exemple: Moteur Nema 17, 1.8° par pas, soit 200 pas pour 360°

Courroie dentée, 2mm d'espace entre dents.

Poulie 20 dents

Opération: (200/40) * 4 (par exemple) = 5

Dans cet exemple, il faut 5 pas des moteurs pour parcourir une distance de 1 millimètre.

Attention, dans GRBL, tout est en millimètres, donc, si vous avez des dimensions en pouces (inches), n'oubliez pas de multiplier par 25,4 avant de faire vos opérations.

Paramètres 130, 131 et 132. Distance maximum à parcourir sur chacun des axes.

Ici, il n'y a pas de calcul à faire, il suffit d'entrer en millimètres la longueur de coupe sur cet axe. En théorie, et en l'absence de fausses manipulations, si ces distances sont entrées correctement, les détecteurs de fin de course devraient être inutiles.

Paramètres 110, 111 et 112. Vitesse maximum sur X, Y et Z.

Ici on entre plus dans le domaine de l'expérimentation personnelle que dans le purement rationnel. Il s'agit ici de vitesse de déplacement maximum, HORS TRAVAIL, c'est à dire pour se repositionner, et pas pour couper. Il y a de nombreuses discussions sur les forums à ce sujet, et nombreux sont ceux qui soutiennent que des valeurs de 12 000 à 14 000 sont parfaitement acceptables, alors que d'autres pensent que 5 000 est bien suffisant. Tout dépend de votre mécanique ET de votre électronique. Pour commencer, une valeur de 8 000 me semble correcte, et la moitié, 4 000, pour l'axe des Z. Expérimentez avec des commandes manuelles pour voir comment votre machine se comporte.

Paramètres 24, 25, 26 et 27. Homing. Comme dans E.T., "maison"!

La procédure de homing (retour à la maison, plus techniquement, point 0, où X= 0, Y = 0 et Z = 0), est importante, et pas le plus simple réglage à effectuer, mais il n'est pas obligatoire, simplement il faudra, s'il n'est pas utilisé, que tous vos fichiers G-Code soient générés pour exprimer des coordonnées relatives, et pas absolues, un concept sur lequel nous reviendrons plus tard. Tout d'abord, ces paramètres ne seront pris en compte que si le paramètre 22 est à 1 (activé), et il faut que vous ayez câblé vos fins de course.

La procédure de Homing va déplacer le portique sur X et Y, d'abord relativement vite, jusqu'à ce qu'il touche les fins de course, puis plus lentement pour détecter le point exact où l'interrupteur se déclenche, en effectuant des mouvements lents de va et vient.

- Le **paramètre 24** correspond à cette vitesse lente. 25 mm/min devrait fonctionner. Le but est d'obtenir une localisation exacte et répétable à chaque fois.
- Le **paramètre 25** correspond à la vitesse de déplacement jusqu'à déclencher les interrupteurs. Une vitesse trop élevée, et votre portique risque de heurter violemment vos fins de course. Une vitesse trop lente, et le processus va prendre trop de temps. Expérimentez, 500 à 600 mm/min sont de bonnes valeurs de départ.
- Le **paramètre 26** est la valeur en millisecondes pendant laquelle le signal électrique de l'interrupteur peut "rebondir" lorsqu'il est déclenché. Ce bruit peut (et devrait!) être filtré électroniquement, mais GRBL utilise ce court délai pour laisser le temps au signal de se stabiliser.
- Le **paramètre 27** est la valeur de laquelle le processus de homing va se décaler des fins de course, de manière à éviter un déclenchement accidentel des fins de course, une fois que le processus à été achevé.

Il n'est normalement pas nécessaire de toucher aux autres paramètres, sauf cas très particuliers, sur lesquels nous reviendrons éventuellement à l'occasion.

Comment changer les paramètres?

Il suffit de taper `$nombre du paramètre = nouvelle valeur`, par exemple `$110=10000`. Faites à nouveau `$$` pour vérifier les nouvelles valeurs. Comme vous pouvez le voir sur cette copie d'écran les paramètres 110, 111 et 112 ont été modifiés.

Conclusion

J'espère que cet article aura levé certaines difficultés à l'utilisation de Grbl. Bien que ce programme ne nécessite qu'un Arduino Uno, il est assez complet et permet de piloter des machines CNC "comme les grands". Dans un prochaine article, nous évoquerons différents programmes pilotant l'Arduino par le biais d'une interface graphique, et examinerons l'un deux plus en détails, sa version Française devant être disponible sous peu.

[CP_CALCULATED_FIELDS id="6"]

Bien que tout ce qui précède soit relatif à GRBL, un certain nombre de principes et de calculs sont valables pour TOUTES les machine CNC.