PARAMÈTRES DU GRBL

Note : Homing = recherche du point zéro de référence

Grbl's `$x=val` settings and what they mean

REMARQUE : La numérotation des réglages a changé depuis la v0.8c à des fins d'évolutivité.	NOTE : Settings numbering has changed since v0.8c for future-proofing purposes.
$0 – Step pulse, microseconds Les drivers de moteurs pas à pas sont conçus pour une certaine durée minimale d'impulsions de pas. Consultez la fiche de données ou tout simplement essayer quelques numéros. Vous voulez que les impulsions les plus courtes soient reconnues de manière fiable. Si les impulsions sont trop longues, vous pourriez rencontrer des erreurs lors de l'exécution à vitesse élevée et des impulsions trop élevées commencent à se chevaucher. Nous recommandons une valeur autour de 10 microsecondes, ce qui est la valeur par défaut.	**$0 – Step pulse, microseconds** Stepper drivers are rated for a certain minimum step pulse length. Check the data sheet or just try some numbers. You want the shortest pulses the stepper drivers can reliably recognize. If the pulses are too long, you might run into trouble when running the system at very high feed and pulse rates, because the step pulses can begin to overlap each other. We recommend something around 10 microseconds, which is the default value.
$1 - Step idle delay, msec Chaque fois que vos moteurs exécutent un mouvement et viennent à un arrêt, Grbl va retarder la désactivation des moteurs par cette valeur ou vous pouvez toujours garder vos axes en fonction (alimentés de manière à maintenir la position) en définissant cette valeur au maximum de 255 millisecondes. Encore une fois, juste pour répéter, vous pouvez garder tous les axes toujours activés en positionnant $1 = 255. Le temps idle de verrouillage est le temps que Grbl gardera les moteurs pas à pas verrouillés avant de les désactiver. Selon le système, vous pouvez définir cette valeur à zéro et ainsi désactiver l'action. Sur d'autres, vous devrez peut-être mettre 25-50 millisecondes pour vous assurer que vos axes parviennent à un arrêt complet avant de désactiver. Ceci pour tenir compte des moteurs pas à pas de machines qui ne peuvent pas rester inactives sur de longues périodes. Aussi, gardez à l'esprit que certains pilotes de moteur pas à pas ne se souviennent pas à quel micro pas ils se sont arrêtés, alors quand vous réactivez, vous pouvez avoir des pertes de pas. Dans ce cas, il suffit de garder vos moteurs activés par $1 = 255.	**$1 - Step idle delay, msec** Every time your steppers complete a motion and come to a stop, Grbl will delay disabling the steppers by this value. **OR**, you can always keep your axes enabled (powered so as to hold position) by setting this value to the maximum 255 milliseconds. Again, just to repeat, you can keep all axes always enabled by setting $1=255. The stepper idle lock time is the time length Grbl will keep the steppers locked before disabling. Depending on the system, you can set this to zero and disable it. On others, you may need 25-50 milliseconds to make sure your axes come to a complete stop before disabling. This is to help account for machine motors that do not like to be left on for long periods of time without doing something. Also, keep in mind that some stepper drivers don't remember which micro step they stopped on, so when you re- enable, you may witness some 'lost' steps due to this. In this case, just keep your steppers enabled via $1=255.
$2 – Step port invert mask:binary Ce paramètre inverse le signal d'impulsions de pas. Par défaut, un signal de pas commence à normalement BAS et va vers le niveau HAUT lors d'un événement d'impulsions de pas. Après un temps d'impulsions de pas fixé par $0, la broche se remet à un niveau BAS, jusqu'à l'impulsion du pas suivant. Quand inversé, le comportement de l'impulsion de pas passe de normalement HAUT à BAS et le retour à HAUT. La plupart des utilisateurs n'auront pas besoin de ce paramètre mais cela peut être utile pour certains pilotes de CNC à moteurs pas à pas qui ont des exigences particulières. Par exemple, un retard artificiel entre la broche de direction et l'impulsion de pas peut être créé en inversant la pin de pas. Ce paramètre de masque est une valeur qui stocke les axes à inverser avec des données binaires. Vous n'avez vraiment pas besoin de comprendre complètement comment cela fonctionne. Il vous suffit de saisir la valeur des paramètres pour les axes que vous souhaitez inverser. Par exemple, si vous voulez inverser les axes X et Z, vous enverriez $ = 5 à Grbl et maintenant devriez lire $ = 5 (valeur masque step : 00000101).	**$2 – Step port invert mask:binary** This setting inverts the step pulse signal. By default, a step signal starts at normal-low and goes high upon a step pulse event. After a step pulse time set by $0, the pin resets to low, until the next step pulse event. When inverted, the step pulse behavior switches from normal-high, to low during the pulse, and back to high. Most users will not need to use this setting, but this can be useful for certain CNC-stepper drivers that have peculiar requirements. For example, an artificial delay between the direction pin and step pulse can be created by inverting the step pin. This invert mask setting is a value which stores the axes to invert as bit flags. You really don't need to completely understand how it works. You simply need to enter the settings value for the axes you want to invert. For example, if you want to invert the X and Z axes, you'd send $2=5 to Grbl and the setting should now read $2=5 (step port invert mask:00000101).

Setting Value	Mask	Invert X	Invert Y	Invert Z
0	00000000	N	N	N
1	00000001	Y	N	N
2	00000010	N	Y	N
3	00000011	Y	Y	N
4	00000100	N	N	Y
5	00000101	Y	N	Y
6	00000110	N	Y	Y
7	00000111	Y	Y	Y

Setting Value	Mask	Invert X	Invert Y	Invert Z
0	00000000	N	N	N
1	00000001	Y	N	N
2	00000010	N	Y	N
3	00000011	Y	Y	N
4	00000100	N	N	Y
5	00000101	Y	N	Y
6	00000110	N	Y	Y
7	00000111	Y	Y	Y

$3 – Direction port invert mask:binary
Ce paramètre inverse le signal de direction pour chaque axe. Par défaut, Grbl suppose que les axes se déplacent dans une direction positive lorsque le niveau de pin de direction est BAS et en sens négatif lorsque le niveau de la broche est HAUT. Souvent, les axes ne se déplacent pas de cette façon sur certaines machines. Ce paramètre va inverser le signal de l'axe de direction de ces axes qui se déplacent dans le sens opposé.
Ce paramètre de masque fonctionne exactement comme le masque de port step et enregistre, par les données binaires quels axes sont à inverser. Pour configurer ce paramètre, il vous suffit d'envoyer la valeur pour les axes que vous souhaitez inverser. Utilisez le tableau ci-dessus. Par exemple, si vous voulez inverser la direction de l'axe Y uniquement, vous devez envoyer $3 = 2 à Grbl et ainsi devriez maintenant lire $3 = 2 (dir masque port invert : 00000010).

$3 – Direction port invert mask:binary
This setting inverts the direction signal for each axis. By default, Grbl assumes that the axes move in a positive direction when the direction pin signal is low, and a negative direction when the pin is high. Often, axes don't move this way with some machines. This setting will invert the direction pin signal for those axes that move the opposite way.

This invert mask setting works exactly like the step port invert mask and stores which axes to invert as bit flags. To configure this setting, you simply need to send the value for the axes you want to invert. Use the table above. For example, if want to invert the Y axis direction only, you'd send $3=2 to Grbl and the setting should now read $3=2 (dir port invert mask : 00000010)

$4 - Step enable invert, bool
Par défaut, la broche d'activation est HAUT pour désactiver et BAS pour valider. Si votre installation a besoin du contraire, inverser simplement la broche d'activation en tapant $4 = 1. Désactiver avec $4 = 0. (Peut nécessiter redémarrage pour charger le changement).

$4 - Step enable invert, bool
By default, the stepper enable pin is high to disable and low to enable. If your setup needs the opposite, just invert the stepper enable pin by typing $4=1. Disable with $4=0. (May need a power cycle to load the change.)

$5 - Limit pins invert, bool
Par défaut, les repères de la course sont normalement HAUT avec la résistance de pull-up interne de l'Arduino. Quand une limite est atteinte le niveau de la broche est BAS, Grbl interprète cela comme déclenchée. Pour le comportement opposé, juste inverser les repères de la limite en tapant $5 = 1. Désactiver avec $5 = 0. Vous aurez peut-être besoin d'un redémarrage pour charger la modification.
NOTE : Si vous inversez les détections de vos limites, vous aurez besoin d'une résistance pull-down externe pour tous les repères de la course pour éviter de surcharger les broches avec le courant et les détruire.

$5 - Limit pins invert, bool
By default, the limit pins are held normally-high with the Arduino's internal pull-up resistor. When a limit pin is low, Grbl interprets this as triggered. For the opposite behavior, just invert the limit pins by typing $5=1. Disable with $5=0. You may need a power cycle to load the change.

NOTE: If you invert your limit pins, you will need an external pull-down resistor wired in to all of the limit pins to prevent overloading the pins with current and frying them.

$6 - Probe pin invert, bool
Par défaut, la pin probe est maintenue normalement HAUT par la résistance de pull-up interne de l' Arduino. Lorsque le capteur fourni le niveau BAS à la broche , Grbl interprète cela comme déclenchée. Pour le comportement opposé, juste inverser la broche de la sonde en tapant $6 = 1. Désactiver avec $6 = 0. Vous aurez peut-être besoin d'un redémarrage pour valider le changement.
NOTE : Si vous inversez le signal de votre tige de palpage, vous aurez besoin d'une résistance pull-down externe câblée sur la broche de la sonde pour éviter la surcharge et éviter des dommages.

$6 - Probe pin invert, bool
By default, the probe pin is held normally-high with the Arduino's internal pull-up resistor. When the probe pin is low, Grbl interprets this as triggered. For the opposite behavior, just invert the probe pin by typing $6=1. Disable with $6=0. You may need a power cycle to load the change.

NOTE : If you invert your probe pin, you will need an external pull-down resistor wired in to the probe pin to prevent overloading it with current and frying it.

$10 - Status report mask:binary
Ce paramètre détermine ce que Grbl doit donner en temps réel comme rapport de situation lorsqu'un '?' est envoyé. Par défaut, Grbl renverra son état de marche (ne peut pas être désactivé), la position de la machine et la position de travail (position de la machine avec les décalages de coordonnées

$10 - Status report mask:binary
This setting determines what Grbl real-time data it reports back to the user when a '?' status report is sent. By default, Grbl will send back its running state (can't be turned off), machine position, and work position (machine position with coordinate offsets and other offsets applied).

et d'autres compensations appliquées). Trois fonctions de rapport supplémentaires sont disponibles et qui sont utiles pour les interfaces ou les utilisateurs qui initialisent leurs machines, qui incluent le tampon série RX, l'utilisation du tampon de bloc de planificateur et les états des pins de limite (BAS ou HAUT, présentés dans l'ordre ZYX).

Pour les définir, utilisez le tableau ci-dessous afin de déterminer quelles sont les données que vous souhaitez que Grbl renvoie. Sélectionnez les types de rapports que vous aimeriez voir dans les rapports de situation et d'ajouter leurs valeurs. Ceci est la valeur que vous utilisez pour envoyer à Grbl. Par exemple, si vous avez besoin des positions de la machine et de travail, ajoutez les valeurs 1 et 2 et envoyer à Grbl $10 = 3 pour le définir ou si vous avez besoin de la position de la machine seule et l'état de la broche de course, ajoutez les valeurs 1 et 16 et envoyer l $10 = 17 à Grbl.

En général, conservez ces données d'état en temps réel à un minimum car il faut des ressources pour imprimer et envoyer ces données à grande vitesse.
Par exemple, les informations de pins de limite sont généralement nécessaires uniquement lorsque les utilisateurs mettent leur machine en place. Ensuite, il est recommandé de le désactiver car il n'est pas très utile une fois que vous aurez tout compris.

Report Type	Value
Machine Position	1
Work Position	2
Planner Buffer	4
RX Buffer	8
Limit Pins	16

Three additional reporting features are available that are useful for interfaces or users setting up their machines, which include the serial RX buffer, planner block buffer usage, and limit pin states (as high or low, shown in the order ZYX).

To set them, use the table below to determine what data you'd like Grbl to send back. Select the report types you'd like to see in the status reports and add their values together. This is the value you use to send to Grbl. For example, if you need machine and work positions, add the values 1 and 2 and send Grbl $10=3 to set it. Or, if you need machine position only and limit pin state, add the values 1 and 16 and send Grbl $10=17.

In general, keep this real-time status data to a minimum, since it takes resources to print and send this data back at a high rate.
For example, limit pins reporting is generally only needed when users are setting up their machine. Afterwards, it's recommended to disable it, as it isn't very useful once you've got everything figured out.

Report Type	Value
Machine Position	1
Work Position	2
Planner Buffer	4
RX Buffer	8
Limit Pins	16

$11 - Junction deviation, mm
L'écart de jonction est utilisé par le gestionnaire d'accélération pour déterminer à quelle vitesse il peut se déplacer à travers les jonctions de segments de ligne d'un chemin de programme G- code. Par exemple, si le chemin G-code a un tour de 10 degrés et la machine se déplace à pleine vitesse, ce paramètre permet de déterminer de combien la machine doit ralentir pour aller en toute sécurité vers le coin sans perdre de pas.
Nos calculs sont un peu compliqués mais en général, des valeurs plus élevées donnent un mouvement plus rapide dans les courbes, tout en augmentant le risque de perdre des pas et le positionnement. Des valeurs inférieures rendent le gestionnaire d'accélération plus prudent et mèneront plus lentement à une courbe. Donc, si vous avez des problèmes là où votre machine essaie de prendre un coin trop vite, diminuer cette valeur pour le faire ralentir en entrée de virage. Si vous voulez que votre machine se déplace plus rapidement à travers les jonctions, augmentez cette valeur pour l'accélérer. Pour les curieux, cliquez ce lien pour lire les propos de l'algorithme de courbes Grbl, qui représente à la fois la vitesse et l'angle de jonction avec une méthode très simple et efficace.

$11 - Junction deviation, mm
Junction deviation is used by the acceleration manager to determine how fast it can move through line segment junctions of a G-code program path. For example, if the G-code path has a sharp 10 degree turn coming up and the machine is moving at full speed, this setting helps determine how much the machine needs to slow down to safely go through the corner without losing steps.
How we calculate it is a bit complicated, but, in general, higher values gives faster motion through corners, while increasing the risk of losing steps and positioning. Lower values makes the acceleration manager more careful and will lead to careful and slower cornering. So if you run into problems where your machine tries to take a corner too fast, *decrease* this value to make it slow down when entering corners. If you want your machine to move faster through junctions, *increase* this value to speed it up. For curious people, hit this link to read about Grbl's cornering algorithm, which accounts for both velocity and junction angle with a very simple, efficient, and robust method.

$12 – Arc tolerance, mm
G2 / G3 de Grbl rend cercles, arcs et hélices en les subdivisant en petites lignes minuscules, tels que la précision de l'arc tracé ne soit jamais en dessous de cette valeur. Vous n'aurez probablement jamais besoin de

$12 – Arc tolerance, mm
Grbl renders G2/G3 circles, arcs, and helices by subdividing them into teeny tiny lines, such that the arc tracing accuracy is never below this value. You will probably never need to adjust this setting, since 0.002mm is well below the

modifier ce paramètre, puisque 0.002 mm est bien en dessous de la précision de la plupart des machines CNC. Mais si vous trouvez que vos cercles sont trop bruts ou que le traçage de l'arc s'effectue lentement, modifiez ce paramètre. Des valeurs inférieures donnent une plus grande précision, mais peut conduire à des problèmes de performance en surchargeant Grbl avec trop de lignes minuscules. Alternativement, des valeurs plus élevées travaillent avec une précision inférieure mais peuvent accélérer les performances de l'arc depuis Grbl qui a moins de lignes à traiter. Pour les curieux, la tolérance de l'arc est définie comme la distance perpendiculaire maximale d'un segment de ligne avec ses points d'extrémité se trouvant sur l'arc, c'est à dire une corde. Avec une géométrie de base, on résout, pour la longueur des segments de ligne à tracer, l'arc qui répond à ce paramètre. La modélisation des arcs est grande de cette manière, parce que les segments d'arc ajustent automatiquement l'échelle de longueur pour assurer les performances de l'arc optimal de traçage, tout en ne perdant jamais de précision.	accuracy of most all CNC machines. But if you find that your circles are too crude or arc tracing is performing slowly, adjust this setting. Lower values give higher precision but may lead to performance issues by overloading Grbl with too many tiny lines. Alternately, higher values traces to a lower precision, but can speed up arc performance since Grbl has fewer lines to deal with. For the curious, arc tolerance is defined as the maximum perpendicular distance from a line segment with its end points lying on the arc, aka a chord. With some basic geometry, we solve for the length of the line segments to trace the arc that satisfies this setting. Modeling arcs in this way is great, because the arc line segments automatically adjust and scale with length to ensure optimum arc tracing performance, while never losing accuracy.
$13 - Report inches, bool Grbl a une fonctionnalité de rapports de positionnement en temps réel pour fournir une rétroaction à l'utilisateur sur la position exacte de la machine à cet instant, ainsi que, pour coordonner les paramètres de compensation et de sondage. Par défaut, il est mis pour signaler en mm mais en envoyant une commande de $13 = 1, vous envoyez ce flag booléen à true et ces fonctions de reports seront à présent en pouces. $13 = 0 retour en mm.	**$13 - Report inches, bool** Grbl has a real-time positioning reporting feature to provide a user feedback on where the machine is exactly at that time, as well as, parameters for coordinate offsets and probing. By default, it is set to report in mm, but by sending a $13=1 command, you send this boolean flag to true and these reporting features will now report in inches. $13 = 0 to set back to mm.
$20 - Soft limits, bool Soft limits est un dispositif de sécurité pour empêcher votre machine de se mouvoir trop loin et au-delà des limites de déplacement et éviter la casse. Il fonctionne en connaissant les limites maximales de déplacement pour chaque axe et où Grbl est en coordonnées machine. Chaque fois qu'une nouvelle information G-code est envoyée à Grbl, il vérifie si vous avez accidentellement dépassé les limites de la machine. Si vous êtes dans ce cas, Grbl stoppera, y compris l'arrêt de la broche et du liquide de refroidissement, puis règle l'alarme du système indiquant le problème. La position de la machine sera conservée car l'arrêt n'est pas du à un arrêt forcé immédiat comme des limites strictes. REMARQUE: Soft limits nécessite que homing soit activé et que les paramètres de déplacement maximaux d'axe soient activés, parce que Grbl a besoin de savoir où il est. $20 = 1 pour activer et $20 = 0 pour désactiver.	**$20 - Soft limits, bool** Soft limits is a safety feature to help prevent your machine from traveling too far and beyond the limits of travel, crashing or breaking something expensive. It works by knowing the maximum travel limits for each axis and where Grbl is in machine coordinates. Whenever a new G-code motion is sent to Grbl, it checks whether or not you accidentally have exceeded your machine space. If you do, Grbl will issue an immediate feed hold wherever it is, shutdown the spindle and coolant, and then set the system alarm indicating the problem. Machine position will be retained afterwards, since it's not due to an immediate forced stop like hard limits. NOTE: Soft limits requires homing to be enabled and accurate axis maximum travel settings, because Grbl needs to know where it is. $20=1 to enable, and $20=0 to disable.
$21 - Hard limits, bool Hard limits fonctionne fondamentalement de la même manière que Soft limits, mais utilise des commutateurs physiques. Fondamentalement, vous câblez quelques interrupteurs (mécaniques, magnétiques ou optiques) près de la fin de course de chaque axe si jamais vous sentez qu'il pourrait y avoir des problèmes si votre programme s'approcherait de trop des limites fixées. Lorsque l'interrupteur s'active, il va immédiatement arrêter tout mouvement, l'arrêt du refroidissement et de la broche (si connecté) et passer en mode d'alarme, ce qui vous oblige à vérifier votre machine et de tout réinitialiser. Pour utiliser les Hard limits avec Grbl, les pins de fin de course sont tenus HAUT avec une résistance interne pull-up donc tout ce que vous avez à faire est de brancher un fil dans un commutateur normalement ouvert entre la broche et la masse et mettre Hard limits à $21 = 1. (Désactiver avec $21 = 0). Nous vous conseillons fortement de prendre les mesures de prévention d'interférences électriques. Si vous voulez une limite pour les deux extrémités de course d'un des axes, il faudra brancher deux interrupteurs en parallèle entre la broche et la masse, si l'un des deux s'enclenche, il	**$21 - Hard limits, bool** Hard limit work basically the same as soft limits, but use physical switches instead. Basically you wire up some switches (mechanical, magnetic, or optical) near the end of travel of each axes, or where ever you feel that there might be trouble if your program moves too far to where it shouldn't. When the switch triggers, it will immediately halt all motion, shutdown the coolant and spindle (if connected), and go into alarm mode, which forces you to check your machine and reset everything. To use hard limits with Grbl, the limit pins are held high with an internal pull-up resistor, so all you have to do is wire in a normally-open switch with the pin and ground and enable hard limits with $21=1. (Disable with $21=0.). We strongly advise taking electric interference prevention measures. If you want a limit for both ends of travel of one axes, just wire in two switches in parallel with the pin and ground, so if either one of them trips, it triggers the hard limit.

déclenche la Hard limits. Gardez à l'esprit qu'un événement de limite matérielle est considéré comme un événement critique où les moteurs pas à pas s'arrêtent immédiatement et auront probablement perdus des pas. Grbl n'a pas de commentaires sur la situation, de sorte qu'il ne peut garantir et qu'il n'a aucune idée où il est. Donc, si une Hard limits est déclenchée, Grbl ira dans un mode boucle d'alarme infinie, vous donnant une obligation de vérifier votre machine et vous obligeant à réinitialiser Grbl. Rappelez-vous que c'est une fonction de sécurité.	Keep in mind, that a hard limit event is considered to be critical event, where steppers immediately stop and will have likely have lost steps. Grbl doesn't have any feedback on position, so it can't guarantee it has any idea where it is. So, if a hard limit is triggered, Grbl will go into an infinite loop ALARM mode, giving you a chance to check your machine and forcing you to reset Grbl. Remember it's a purely a safety feature.
$22 - Homing cycle, bool Ah! homing. Pour ceux qui sont juste initié à la CNC, le cycle homing est utilisé pour localiser avec exactitude une position connue et cohérente sur une machine, chaque fois que vous démarrez votre Grbl entre les sessions. En d'autres termes, vous savez exactement où vous êtes à tout moment, à chaque fois. Disons que vous commencez l'usinage d'une pièce ou vous êtes sur le point de commencer la prochaine étape et une panne de courant redémarre Grbl et Grbl n'a aucune idée de l'endroit où il se trouve. Vous êtes incapable de déterminer où vous en êtes avec la tâche. Si vous avez un Homing, vous avez toujours le point de référence zéro machine pour localiser tout ce que vous avez à faire à partir de là, de lancer le cycle de homing et de reprendre là où vous vous étiez arrêté. Pour mettre en place le cycle de homing pour Grbl, vous devez disposer d'interrupteurs de fin de course fixés solidement pour avoir un point de référence bien précis. Habituellement, ils sont configurés dans le point le plus éloigné de + x, + y, + z de chaque axe. Câbler vos fins de course entre les pins fin de course et la masse, tout comme avec les hard limits, afin de permettre la localisation de Homing. Vous pouvez utiliser vos interrupteurs de limite pour les deux hard limits et homing. Ils jouent le même rôle. Par défaut, le cycle de homing de Grbl déplace l'axe Z positif en premier pour dégager l'espace de travail, puis déplace à la fois X et Y en même temps dans le sens positif. Pour configurer la façon dont votre cycle de homing se comporte, il y a des réglages Grbl plus en bas de la page décrivant ce qu'ils font (et la compilation des options) Aussi, encore une chose à noter, lorsque homing est activée. Grbl verrouille toutes les commandes G-code jusqu'à ce que vous effectuiez un cycle de homing. Cela signifie qu'aucun mouvement d'axes n'est effectué, à moins que le verrouillage ne soit désactivé ($X) mais plus sur cela plus tard. La plupart, sinon tous les contrôleurs CNC, font quelque chose de semblable car ils ont la plupart du temps un dispositif de sécurité pour empêcher les utilisateurs de faire une erreur de positionnement. C'est tellement vite arrivé et ennuyeux si une partie de la mécanique est détruite. Si vous trouvez des anomalies ou des bugs, s'il vous plaît, faites-nous le savoir et nous allons essayer d'y remédier. NOTE : Consultez config.h pour plus d'options de homing pour les utilisateurs avancés. Vous pouvez désactiver le lock-out de homing au démarrage, configurer le déplacement des axes, leur ordre et bien plus encore.	**$22 - Homing cycle, bool** Ahh, homing. For those just initiated into CNC, the homing cycle is used to accurately and precisely locate a known and consistent position on a machine every time you start up your Grbl between sessions. In other words, you know exactly where you are at any given time, every time. Say you start machining something or are about to start the next step in a job and the power goes out, you re- start Grbl and Grbl has no idea where it is. You're left with the task of figuring out where you are. If you have homing, you always have the machine zero reference point to locate from, so all you have to do is run the homing cycle and resume where you left off. To set up the homing cycle for Grbl, you need to have limit switches in a fixed position that won't get bumped or moved, or else your reference point gets messed up. Usually they are setup in the farthest point in +x, +y, +z of each axes. Wire your limit switches in with the limit pins and ground, just like with the hard limits, and enable homing. If you're curious, you can use your limit switches for both hard limits AND homing. They play nice with each other. By default, Grbl's homing cycle moves the Z-axis positive first to clear the workspace and then moves both the X and Y-axes at the same time in the positive direction. To set up how your homing cycle behaves, there are more Grbl settings down the page describing what they do (and compile-time options as well.) Also, one more thing to note, when homing is enabled. Grbl will lock out all G-code commands until you perform a homing cycle. Meaning no axes motions, unless the lock is disabled ($X) but more on that later. Most, if not all CNC controllers, do something similar, as it is mostly a safety feature to prevent users from making a positioning mistake, which is very easy to do and be saddenee when a mistake ruins a part. If you find this annoying or find any weird bugs, please let us know and we'll try to work on it so everyone is happy. :) NOTE : Check out config.h for more homing options for advanced users. You can disable the homing lockout at startup, configure which axes move first during a homing cycle and in what order, and more.
$23 - Homing dir invert mask, int:binary Par défaut, Grbl suppose que vos fins de course de homing sont dans le sens positif, le premier déplacement de l'axe z positif, alors axes XY positif avant d'essayer de localiser précisément le zéro machine en allant d'avant en arrière lentement autour de l'interrupteur.	**$23 - Homing dir invert mask, int:binary** By default, Grbl assumes your homing limit switches are in the positive direction, first moving the z-axis positive, then the x-y axes positive before trying to precisely locate machine zero by going back and forth slowly around the switch.

Si votre machine dispose d'un interrupteur de fin de course dans le sens négatif, le masque de direction homing peut inverser la direction des axes. Il fonctionne exactement comme l'inverseur du port de pas et du port de masques, où tout ce que vous avez à faire est d'envoyer la valeur dans le tableau pour indiquer quels axes vous voulez inverser et faire la recherche dans la direction opposée.	If your machine has a limit switch in the negative direction, the homing direction mask can invert the axes' direction. It works just like the step port invert and direction port invert masks, where all you have to do is send the value in the table to indicate what axes you want to invert and search for in the opposite direction.
$24 - Homing feed, mm/min Le premier cycle de homing recherche les fins de course à une vitesse plus élevée et après les avoir trouvés, se déplace plus lentement pour détecter l'emplacement précis du zéro machine. Homing est plus lent que la vitesse de déplacement. Réglez cette valeur quelconque à un taux qui fournit à la machine la localisation précise du zéro.	**$24 - Homing feed, mm/min** The homing cycle first searches for the limit switches at a higher seek rate, and after it finds them, it moves at a slower feed rate to home into the precise location of machine zero. Homing feed rate is that slower feed rate. Set this to whatever rate value that provides repeatable and precise machine zero locating.
$25 - Homing seek, mm/min Homing seek est les mm/min de recherche sur le cycle de prise d'origine ou la vitesse à laquelle il tente d'abord de trouver les fins de course. Réglez ce taux afin d'arriver au fin de course dans un temps assez court mais sans tout casser si c'est trop rapide.	**$25 - Homing seek, mm/min** Homing seek rate is the homing cycle search rate, or the rate at which it first tries to find the limit switches. Adjust to whatever rate gets to the limit switches in a short enough time without crashing into your limit switches if they come in too fast.
$26 - Homing debounce, ms Chaque fois qu'un interrupteur déclenche, on peut avoir un bruit électrique / mécanique qui fait «rebondir» le signal HAUT et BAS pour quelques millisecondes avant la stabilisation. Pour résoudre ce problème, vous devez éviter les rebonds du signal, soit par le matériel avec une sorte de conditionneur de signal ou de logiciels avec un court délai pour laisser le temps du rebond et d'arriver avec un signal stabilisé. Grbl effectue un délai court, seulement pendant la localisation du zéro machine. Définissez cette valeur de retard pour que, quel que soit votre commutateur, vous obteniez un homing répétable. Dans la plupart des cas, 5-25 millisecondes est très bien.	**$26 - Homing debounce, ms** Whenever a switch triggers, some of them can have electrical/mechanical noise that actually 'bounce' the signal high and low for a few milliseconds before settling in. To solve this, you need to debounce the signal, either by hardware with some kind of signal conditioner or by software with a short delay to let the signal finish bouncing. Grbl performs a short delay, only homing when locating machine zero. Set this delay value to whatever your switch needs to get repeatable homing. In most cases, 5-25 milliseconds is fine.
$27 - Homing pull-off, mm Dans le cas de partage des mêmes fins de course et de homing, les fins de course ne fonctionnent qu'en homing. En d'autres termes, il aide à prévenir le déclenchement accidentel des limites après un cycle de homing.	**$27 - Homing pull-off, mm** To play nice with the hard limits feature, where homing can share the same limit switches, the homing cycle will move off all of the limit switches by this pull-off travel after it completes. In other words, it helps to prevent accidental triggering of the hard limit after a homing cycle.
$30 – Punch actuator down invert, bool $30 = 0 — Commandé par : +5 V $30 = 1 — Commandé par : 0 V	**$30 – Punch actuator down invert, bool**
$31 – Punch actuator up invert, bool $31 = 0 — Commandé par : +5 V $31 = 1 — Commandé par : 0 V	**$31 – Punch actuator up invert, bool**
$32 – Punch sensor down invert, bool $32 = 0 — Si détecté = +5 V $32 = 1 — Si détecté = 0 V	**$32 – Punch sensor down invert, bool**
$33 – Punch sensor up invert, bool $33 = 0 — Si détecté = +5 V $33 = 1 — Si détecté = 0 V	**$33 – Punch sensor up invert, bool**
$100, $101 and $102 – [X,Y,Z] steps/mm Grbl a besoin de savoir dans quelle mesure chaque étape placera l'outil dans la réalité. Pour calculer pas/ mm pour un axe de votre machine, vous devez savoir : Les mm parcourus par la rotation de votre moteur	**$100, $101 and $102 – [X,Y,Z] steps/mm** Grbl needs to know how far each step will take the tool in reality. To calculate steps/mm for an axis of your machine you need to know : The mm traveled per revolution of your stepper

pas à pas. Cela dépend de vos engrenages d'entraînement des courroies ou du pas des vis. Les pas complets pour un tour de vos moteurs pas à pas (typiquement 200) Les micropas par pas de votre contrôleur (typiquement 1, 2, 4, 8, ou 16). *Astuce : L'utilisation de valeurs élevées de micropas (par exemple, 16) peut réduire votre couple de moteur pas à pas, afin d'utiliser le plus bas qui vous donne la résolution de l'axe désiré et propriétés de fonctionnement confortables.* Les pas/ mm peuvent alors être calculés comme ceci : pas_par_mm = (pas_par_tour * micropas) / mm_par_tour Calculer cette valeur pour chaque axe et écrire ces paramètres dans Grbl.	motor. This is dependent on your belt drive gears or lead screw pitch. The full steps per revolution of your steppers (typically 200) The microsteps per step of your controller (typically 1, 2, 4, 8, or 16). *Tip: Using high microstep values (e.g., 16) can reduce your stepper motor torque, so use the lowest that gives you the desired axis resolution and comfortable running properties.* The steps/mm can then be calculated like this: steps_per_mm = (steps_per_revolution*microsteps)/mm_per_rev Compute this value for every axis and write these settings to Grbl.
$110, $111 and $112 – [X,Y,Z] Max rate, mm/min Ceci définit le déplacement maximal possible de chaque axe. Chaque fois que Grbl prévoit un déplacement, il vérifie si oui ou non le mouvement provoque un quelconque dépassement de la course max de ces différents axes. Si oui, il va ralentir le mouvement pour vous assurer qu'aucun des axes ne dépasse sa limite de déplacement max. Cela signifie que chaque axe dispose de sa propre vitesse indépendante, ce qui est extrêmement utile pour limiter la vitesse de l'axe Z typiquement plus lente. La façon la plus simple pour déterminer ces valeurs est de tester chaque axe l'un après l'autre en augmentant lentement les réglages de déplacement max. Par exemple, pour tester l'axe X, envoyer quelque chose comme G0 X50 à Grbl avec assez de distance de parcours de sorte que l'axe accélère à sa vitesse maximale. Vous saurez que vous avez touché le seuil de taux max lorsque vos moteurs pas à pas décrochent. Ca va faire un peu de bruit, mais ne devrait pas nuire à vos moteurs. Entrez un réglage de 10-20% en dessous de cette valeur, de sorte que vous pouvez tenir compte de l'usure, la friction et la masse de votre pièce / outil. Ensuite, répétez l'opération pour vos autres axes.	**$110, $111 and $112 – [X,Y,Z] Max rate, mm/min** This sets the maximum rate each axis can move. Whenever Grbl plans a move, it checks whether or not the move causes any one of these individual axes to exceed their max rate. If so, it'll slow down the motion to ensure none of the axes exceed their max rate limits. This means that each axis has its own independent speed, which is extremely useful for limiting the typically slower Z-axis. The simplest way to determine these values is to test each axis one at a time by slowly increasing max rate settings and moving it. For example, to test the X-axis, send Grbl something like G0 X50 with enough travel distance so that the axis accelerates to its max speed. You'll know you've hit the max rate threshold when your steppers stall. It'll make a bit of noise, but shouldn't hurt your motors. Enter a setting a 10-20% below this value, so you can account for wear, friction, and the mass of your workpiece/tool. Then, repeat for your other axes.
$120, $121, $122 – [X,Y,Z] Accélération, mm/sec^2 Ceci définit les paramètres d'accélération des axes en mm / seconde / seconde. De manière simpliste, une valeur inférieure soulage et rend Grbl plus lent en mouvement, tandis qu'une valeur plus élevée accélère le mouvement. Tout comme le réglage de la fréquence max, chaque axe dispose de sa propre valeur d'accélération et ils sont indépendants les uns des autres. Cela signifie qu'un mouvement multi-axe ne fera accélérer aussi rapidement que l'axe le plus bas ne peut le faire. Encore une fois, comme le réglage de la fréquence max, la façon la plus simple pour déterminer les valeurs de ce paramètre est de tester individuellement chaque axe de plus en plus petites valeurs jusqu'à ce que le moteur cale. Puis finaliser votre paramètre d'accélération avec une valeur de 10-20% en dessous de cette valeur absolue max. Cela devrait tenir compte de l'usure, la friction et l'inertie de masse. Nous recommandons fortement d'essayer les programmes G-code à blanc avec vos nouveaux paramètres avant de lancer une opération d'usinage. Parfois, le chargement sur votre machine est différent lors du déplacement simultané de plusieurs axes.	**$120, $121, $122 – [X,Y,Z] Acceleration, mm/sec^2** This sets the axes acceleration parameters in mm/second/second. Simplistically, a lower value makes Grbl ease slower into motion, while a higher value yields tighter moves and reaches the desired feedrates much quicker. Much like the max rate setting, each axis has its own acceleration value and are independent of each other. This means that a multi-axis motion will only accelerate as quickly as the lowest contributing axis can. Again, like the max rate setting, the simplest way to determine the values for this setting is to individually test each axis with slowly increasing values until the motor stalls. Then finalize your acceleration setting with a value 10-20% below this absolute max value. This should account for wear, friction, and mass inertia. We highly recommend that you dry test some G-code programs with your new settings before committing to them. Sometimes the loading on your machine is different when moving in all axes together.
$130, $131, $132 – [X,Y,Z] Max travel, mm Ceci définit la course maximale de bout en bout pour chaque axe, en mm. Ceci est utile uniquement si vous avez des limites douces (homing) et permis, comme cela est utilisé	**$130, $131, $132 – [X,Y,Z] Max travel, mm** This sets the maximum travel from end to end for each axis in mm. This is only useful if you have soft limits (and homing) enabled, as this is only used by Grbl's soft limit feature to

uniquement par les fonctions de limite souple de Grbl pour vérifier si vous avez dépassé vos limites de la machine avec une commande de mouvement.	check if you have exceeded your machine limits with a motion command.
Grbl's Other '$' Commands	
Les autres commandes de $ fournissent des contrôles supplémentaires pour l'utilisateur, telles que l'impression des commentaires sur l'état modal actuel de l'analyseur G-code ou l'exécution du cycle de homing. Cette section explique ce que sont ces commandes et comment les utiliser.	The other $ commands provide additional controls for the user, such as printing feedback on the current G-code parser modal state or running the homing cycle. This section explains what these commands are and how to use them.
$# - View gcode parameters Les paramètres G-code stockent les valeurs de décalage pour les coordonnées G54-G59, G28 / G30 travail positions prédéfinies, G92 décalage de coordonnées, des décalages de longueur d'outil et le palpeur (pas officiellement, mais nous avons l'ajouté ici). La plupart de ces paramètres sont directement écrit en mémoire EEPROM à chaque fois qu'ils sont modifiés et sont persistants. Ce qui signifie qu'ils resteront les mêmes, indépendamment de la puissance vers le bas, jusqu'à ce qu'ils soient modifiés explicitement. Les paramètres non persistants, qui ne sont pas conservés lorsque réinitialisation ou redémarrage, sont G92, G43.1 longueur outil de compensations et le G38.2 données du palpeur.	**$# - View gcode parameters** G-code parameters store the coordinate offset values for G54-G59 work coordinates, G28/G30 pre-defined positions, G92 coordinate offset, tool length offsets, and probing (not officially, but we added here anyway). Most of these parameters are directly written to EEPROM anytime they are changed and are persistent. Meaning that they will remain the same, regardless of power-down, until they are explicitly changed. The non-persistent parameters, which will are not retained when reset or power-cycled, are G92, G43.1 tool length offsets, and the G38.2 probing data.
Les coordonnées de travail G54-G59 peuvent être modifiées via la commande G10 L2 Px ou G10 L20 Px défini par la norme gcode NIST et la norme EMC2 (linuxcnc.org). Les positions prédéfinies G28 / G30 peuvent être modifiées par l'intermédiaire du G28.1 et les commandes G30.1, respectivement.	G54-G59 work coordinates can be changed via the G10 L2 Px or G10 L20 Px command defined by the NIST gcode standard and the EMC2 (linuxcnc.org) standard. G28/G30 pre-defined positions can be changed via the G28.1 and the G30.1 commands, respectively.
Lorsque $# est appelé, Grbl répondra avec les décalages enregistrés à partir des coordonnées machine pour chaque système comme suit. TLO dénote la longueur d'outil, et PRB indique les coordonnées du dernier cycle de palpage. `[G54:4.000,0.000,0.000]` `[G55:4.000,6.000,7.000]` `[G56:0.000,0.000,0.000]` `[G57:0.000,0.000,0.000]` `[G58:0.000,0.000,0.000]` `[G59:0.000,0.000,0.000]` `[G28:1.000,2.000,0.000]` `[G30:4.000,6.000,0.000]` `[G92:0.000,0.000,0.000]` `[TLO:0.000,0.000,0.000]` `[PRB:0.000,0.000,0.000]`	When $# is called, Grbl will respond with the stored offsets from machine coordinates for each system as follows. TLO denotes tool length offset, and PRB denotes the coordinates of the last probing cycle. `[G54:4.000,0.000,0.000]` `[G55:4.000,6.000,7.000]` `[G56:0.000,0.000,0.000]` `[G57:0.000,0.000,0.000]` `[G58:0.000,0.000,0.000]` `[G59:0.000,0.000,0.000]` `[G28:1.000,2.000,0.000]` `[G30:4.000,6.000,0.000]` `[G92:0.000,0.000,0.000]` `[TLO:0.000,0.000,0.000]` `[PRB:0.000,0.000,0.000]`
$G - View gcode parser state Cette commande imprime tous les modes de Gcode actifs dans l'analyseur G-Code. Lors de l'envoi de cette commande à Grbl, il répondra avec quelque chose comme : `[G0 G54 G17 G21 G90 G94 M0 M5 M9 T0 S0.0 F500.0]` Ces modes actifs déterminent comment le prochain bloc G-code ou commande sera interprété par l'analyseur G code Pour ceux qui découvrent G-code et l'usinage CNC, modes met l'analyseur dans un état particulier de sorte que vous ne devez pas dire constamment à l'analyseur comment analyser. Ces modes sont organisés en ensembles appelés	**$G - View gcode parser state** This command prints all of the active gcode modes in Grbl's G- code parser. When sending this command to Grbl, it will reply with something like : `[G0 G54 G17 G21 G90 G94 M0 M5 M9 T0 S0.0 F500.0]` These active modes determine how the next G-code block or command will be interpreted by Grbl's G-code parser. For those new to G-code and CNC machining, modes sets the parser into a particular state so you don't have to constantly tell the parser how to parse it. These modes are organized into sets called "modal groups" that cannot be logically

"groupes modaux" qui ne peuvent être logiquement actifs en même temps. Par exemple, le groupe des unités modales définit si votre programme G-code est interprété en pouces ou en millimètres.

Une courte liste des groupes modaux, soutenus par Grbl, est illustrée ci-dessous mais des descriptions plus complètes et détaillées peuvent être trouvées sur le site de LinuxCNC. Les commandes G-code en caractère gras indiquent les modes sur la mise sous tension ou la réinitialisation de Grbl.

Modal Group Meaning	Member Words
Motion Mode	**G0**, G1, G2, G3, G38.2, G38.3, G38.4, G38.5, G80
Coordinate System Select	**G54**, G55, G56, G57, G58, G59
Plane Select	**G17**, G18, G19
Distance Mode	**G90**, G91
Arc IJK Distance Mode	**G91.1**
Feed Rate Mode	G93, **G94**
Units Mode	G20, **G21**
Cutter Radius Compensation	**G40**
Tool Length Offset	G43.1, **G49**
Program Mode	**M0**, M1, M2, M30
Spindle State	M3, M4, **M5**
Coolant State	M7, M8, **M9**

En plus des modes de l'analyseur G-code, Grbl rendra compte sur l'outil actif T, S vitesse de broche et le taux d'alimentation F, tout sera à 0 lors de la réinitialisation. Pour info : ceux-ci ne correspondent pas tout à fait aux groupes modaux mais sont tout aussi importants pour déterminer l'état de l'analyseur.

active at the same time. For example, the units modal group sets whether your G-code program is interpreted in inches or in millimeters.

A short list of the modal groups, supported by Grbl, is shown below, but more complete and detailed descriptions can be found at LinuxCNC's website. The G-code commands in **bold** indicate the default modes upon powering-up Grbl or resetting it.

Modal Group Meaning	Member Words
Motion Mode	**G0**, G1, G2, G3, G38.2, G38.3, G38.4, G38.5, G80
Coordinate System Select	**G54**, G55, G56, G57, G58, G59
Plane Select	**G17**, G18, G19
Distance Mode	**G90**, G91
Arc IJK Distance Mode	**G91.1**
Feed Rate Mode	G93, **G94**
Units Mode	G20, **G21**
Cutter Radius Compensation	**G40**
Tool Length Offset	G43.1, **G49**
Program Mode	**M0**, M1, M2, M30
Spindle State	M3, M4, **M5**
Coolant State	M7, M8, **M9**

In addition to the G-code parser modes, Grbl will report the active T tool number, S spindle speed, and F feed rate, which all default to 0 upon a reset. For those that are curious, these don't quite fit into nice modal groups, but are just as important for determining the parser state.

`$I` - View build info

Ce retour d'impression à l'utilisateur indique la version et la source du code et la date de construction Grbl. Eventuellement, avec $I, on peut également stocker une courte chaîne pour aider à identifier avec quelle machine CNC vous communiquez, si vous avez plusieurs machines pilotées par Grbl. Pour définir cette chaîne, envoyer Grbl $I = xxx, où xxx est votre chaîne de personnalisation qui est inférieure à 80 caractères. La prochaine fois que vous interrogez Grbl avec un $I, Grbl imprimera cette chaîne après la version et la date de construction.

`$I` - View build info

This prints feedback to the user the Grbl version and source code build date.
Optionally, $I can also store a short string to help identify which CNC machine you are communicating with, if you have more than machine using Grbl.
To set this string, send Grbl $I=xxx, where xxx is your customization string that is less than 80 characters. The next time you query Grbl with a $I view build info, Grbl will print this string after the version and build date.

$N - View startup blocks

$Nx sont les blocs de démarrage que Grbl exécute chaque fois que vous allumez Grbl ou réinitialisez Grbl. En d'autres termes, un bloc de démarrage est une ligne de G code que vous pouvez avoir exécuté automatiquement pour configurer votre G- code par défaut modal ou toute autre chose dont vous aurez besoin à chaque fois que vous démarrez votre machine. Grbl peut stocker deux blocs de G code par défaut.

Donc lorsque vous êtes connecté à Grbl, tapez $N, puis « enter ». Grbl doit répondre avec quelque chose de court, comme :

```
$N0=
$N1=
ok
```

$N - View startup blocks

$Nx are the startup blocks that Grbl runs every time you power on Grbl or reset Grbl. In other words, a startup block is a line of G- code that you can have Grbl auto-magically run to set your G-code modal defaults, or anything else you need Grbl to do everytime you start up your machine. Grbl can store two blocks of G-code as a system default.

So, when connected to Grbl, type $N and then enter. Grbl should respond with something short like:

```
$N0=
$N1=
ok
```

Pas grand chose à faire, mais cela signifie simplement qu'il n'y a pas de bloc G-code stocké dans la ligne $N0 de Grbl pour une exécution au démarrage. $N1 est la prochaine ligne à exécuter.	Not much to go on, but this just means that there is no G-code block stored in line $N0 for Grbl to run upon startup. $N1 is the next line to be run.
$Nx=line - Save startup block **IMPORTANT: Soyez très prudent lorsque vous stockez toutes commandes de mouvement (G0 / 1, G2 / 3, G28 / 30) dans les blocs de démarrage. Ces commandes de mouvements se dérouleront chaque fois que vous ré enclenchez Grbl donc si vous avez une situation d'urgence comme l'e-arrêt et reset, un déplacement de bloc de démarrage peut et va probablement aggraver les choses rapidement. Aussi, ne pas placer de commandes qui enregistrent des données sur EEPROM, tels que G10 / G28.1 / G30.1. Cela obligera Grbl à constamment ré-écrire ces données à chaque démarrage et réinitialisation, qui finiront par user l'EEPROM de votre Arduino.** **L'usage typique pour un bloc de démarrage est simplement de définir vos états modaux préférés, tels que G20 mode pouces, toujours par défaut pour un système de coordonnées différent ou pour fournir un moyen d'exécuter une certaine caractéristique unique écrite par l'utilisateur dont il a besoin pour son projet.** Pour définir un bloc de démarrage, tapez $N0 = suivi d'un bloc G-code valide et une entrée. Grbl déroulera le bloc pour vérifier s'il est valide et répondra ensuite avec un OK ou une erreur : S'il y a une erreur, Grbl ne sauvera pas. Par exemple, disons que vous voulez utiliser votre premier bloc de démarrage $N0 à définir vos G codes en mode d'analyseur, comme les coordonnées de travail de G54, G20 en mode pouces. Vous taperez $N0 = G20 G54 G17 avec une entrée et vous devriez voir une réponse 'OK'. Vous pouvez ensuite vérifier si elles ont été stockées en tapant $N et vous devriez maintenant voir une réponse comme $N0 = G20G54G17. Une fois que vous avez un bloc de démarrage de stocké dans l'EEPROM de Grbl, chaque fois que vous démarrez ou réinitialisez, vous verrez votre bloc de démarrage imprimé et une réponse de Grbl pour indiquer si tout est ok. Donc, pour l'exemple précédent, vous verrez : `Grbl 0.9i ['$' for help]` `G20G54G17ok` Si vous avez plusieurs blocs de démarrage en G code, ils sont imprimés à nouveau pour chaque démarrage et si vous souhaitez effacer l'un des blocs de démarrage (par exemple, le bloc 0) tapez $N0 = sans aucune annotation après le signe égal. Ainsi, si vous avez activé homing, les blocs de démarrage seront exécutés immédiatement après ce cycle de homing et non au démarrage.	**$Nx=line - Save startup block** **IMPORTANT: Be very careful when storing any motion (G0/1,G2/3,G28/30) commands in the startup blocks. These motion commands will run everytime you reset or power up Grbl, so if you have an emergency situation and have to e-stop and reset, a startup block move can and will likely make things worse quickly. Also, do not place any commands that save data to EEPROM, such as G10/G28.1/G30.1. This will cause Grbl to constantly re-write this data upon every startup and reset, which will eventually wear out your Arduino's EEPROM.** **Typical usage for a startup block is simply to set your preferred modal states, such as G20 inches mode, always default to a different work coordinate system, or, to provide a way for a user to run some user-written unique feature that they need for their crazy project.** To set a startup block, type $N0= followed by a valid G-code block and an enter. Grbl will run the block to check if it's valid and then reply with an ok or an error: to tell you if it's successful or something went wrong. If there is an error, Grbl will not save it. For example, say that you want to use your first startup block $N0 to set your G-code parser modes like G54 work coordinate, G20 inches mode, G17 XY-plane. You would type $N0=G20 G54 G17 with an enter and you should see an 'ok' response. You can then check if it got stored by typing $N and you should now see a response like $N0=G20G54G17. Once you have a startup block stored in Grbl's EEPROM, everytime you startup or reset you will see your startup block printed back to you and a response from Grbl to indicate if it ran okay. So for the previous example, you'll see : `Grbl 0.9i ['$' for help]` `G20G54G17ok` If you have multiple G-code startup blocks, they will print back to you in order upon every startup. And if you'd like to clear one of the startup blocks, (e.g., block 0) type $N0= without anything following the equal sign. Also, if you have homing enabled, the startup blocks will execute immediately after the homing cycle, not at startup.
$C - Check gcode mode Cette bascule gcode permet à l'analyseur de Grbl de prendre tous les blocs entrants et de les traiter complètement, comme il le ferait en fonctionnement normal mais il ne déplace pas les axes, ignore la position et l'alimentation de la broche et du liquide de refroidissement. Ceci est conçu pour fournir à l'utilisateur un moyen de tester ses nouveaux programmes G-code avec l'analyseur de Grbl et de surveiller d'éventuelles erreurs (et contrôle les violations de limites douces, si activées).	**$C - Check gcode mode** This toggles the Grbl's gcode parser to take all incoming blocks process them completely, as it would in normal operation, but it does not move any of the axes, ignores dwells, and powers off the spindle and coolant. This is intended as a way to provide the user a way to check how their new G-code program fares with Grbl's parser and monitor for any errors (and checks for soft limit violations, if enabled).

Lorsque désactivé, le Grbl effectuera un soft-reset automatique (^ X). Ceci pour deux raisons. Il simplifie un peu la gestion de code mais il empêche également les utilisateurs de démarrer une tâche quand leurs modes G-code ne sont pas ce qu'ils pensent qu'ils sont. Une réinitialisation du système donne toujours un démarrage sain à l'utilisateur.	When toggled off, Grbl will perform an automatic soft-reset (^X). This is for two purposes. It simplifies the code management a bit. But, it also prevents users from starting a job when their G-code modes are not what they think they are. A system reset always gives the user a fresh, consistent start.
$X - Kill alarm lock Le mode d'alarme de Grbl est un état qui signale un problème critique, comme une fin de course activée ou une interruption pendant un cycle ou si Grbl ne connaît pas sa position. Par défaut, si vous avez un homing d'activé à la mise sous tension de l'Arduino, Grbl passe à l'état d'alarme, car il ne connaît pas sa position. Le mode d'alarme verrouille toutes les commandes G-code jusqu'à ce que le cycle de homing, le '$H' a été effectué ou si un utilisateur a besoin de déplacer les axes sur leurs fins de course, par exemple, le verrouillage de kill alarme '$X' remplace les fins de course et permet aux fonctions G-code de fonctionner à nouveau. Mais avancer avec précaution !! Cela ne devrait être utilisé que dans des situations d'urgence. La position a probablement été perdue et Grbl n'est peut-être pas là où vous pensez qu'il soit. Donc, il est conseillé d'utiliser le mode incrémental G91, de faire des déplacements courts. Ensuite, effectuez un cycle de homing ou réinitialiser immédiatement après.	**$X - Kill alarm lock** Grbl's alarm mode is a state when something has gone critically wrong, such as a hard limit or an abort during a cycle, or if Grbl doesn't know its position. By default, if you have homing enabled and power-up the Arduino, Grbl enters the alarm state, because it does not know its position. The alarm mode will lock all G-code commands until the '$H' homing cycle has been performed. Or if a user needs to override the alarm lock to move their axes off their limit switches, for example, '$X' kill alarm lock will override the locks and allow G-code functions to work again. But, tread carefully!! This should only be used in emergency situations. The position has likely been lost, and Grbl may not be where you think it is. So, it's advised to use G91 incremental mode to make short moves. Then, perform a homing cycle or reset immediately afterwards.
$H - Run homing cycle Cette commande est la seule façon pour effectuer le cycle de Homing dans Grbl. Certains autres contrôleurs de mouvements désignent une commande spéciale de G-code et exécutent un cycle de Homing mais cela est inexact selon les normes G-code. Homing est une commande complètement séparée et gérée par le contrôleur. CONSEIL : Après l'exécution d'un cycle de Homing, mettez plutôt le jogging manuellement dans une position au milieu de votre espace de travail. Vous pouvez placer un G28 ou G30 pour prédéfinir la position pour être à votre position de post-homing, la plus proche de l'endroit où vous serez en usinage. Pour régler cela, vous devez d'abord vous rappelez où la machine se déplace après le homing. Taper G28.1 (ou G30.1) après avoir stocké cette position dans Grbl. Alors après homing '$H', vous pouvez simplement saisir 'G28' (ou 'G30') et il va se déplacer automatiquement. En général, je voudrais juste déplacer l'axe XY du centre et ignorer l'axe Z. Cela garantit qu'il n'y a pas d'interférences avec l'outil dans la broche et que rien ne se coïnce.	**$H - Run homing cycle** This command is the only way to perform the homing cycle in Grbl. Some other motion controllers designate a special G-code command to run a homing cycle, but this is incorrect according to the G-code standards. Homing is a completely separate command handled by the controller. TIP : After running a homing cycle, rather jogging manually all the time to a position in the middle of your workspace volume. You can set a G28 or G30 pre-defined position to be your post-homing position, closer to where you'll be machining. To set these, you'll first need to jog your machine to where you would want it to move to after homing. Type G28.1 (or G30.1) to have Grbl store that position. So then after '$H' homing, you could just enter 'G28' (or 'G30') and it'll move there auto-magically. In general, I would just move the XY axis to the center and leave the Z-axis up. This ensures that there isn't a chance the tool in the spindle will interfere and that it doesn't catch on anything.
$RST=$, $RST=#, and $RST=*- Restore Grbl settings and data to defaults Ces commandes ne sont pas répertoriées dans les messages d'aide de Grbl$ mais sont disponibles pour permettre aux utilisateurs de restaurer des parties ou l'ensemble des données de l'EEPROM de Grbl. Remarque : Grbl se réinitialise automatiquement après l'exécution de l'une de ces commandes pour assurer que le système soit initialisé correctement. • $RST = $: Efface et restaure les paramètres Grbl $$ par défaut, qui sont définis par le fichier de paramètres par défaut utilisé lors de la compilation de Grbl. Souvent, les équipementiers vont construire leurs machines avec leurs paramètres recommandés spécifiques. Cela fournit aux utilisateurs et aux équipementiers un moyen rapide d'obtenir un retour à la case départ, si quelque chose a mal tourné ou si un utilisateur veut recommencer. • $RST = # : Efface et met à zéro toutes les coordonnées G54-G59 et les coordonnées G28/30 mémorisées dans l'EEPROM. Ce sont généralement les valeurs observées	**$RST=$, $RST=#, and $RST=*- Restore Grbl settings and data to defaults** These commands are not listed in the main Grbl $ help message, but are available to allow users to restore parts of or all of Grbl's EEPROM data. Note: Grbl will automatically reset after executing one of these commands to ensure the system is initialized correctly. • $RST = $: Erases and restores the $$ Grbl settings back to defaults, which is defined by the default settings file used when compiling Grbl. Often OEMs will build their Grbl firmwares with their machine-specific recommended settings. This provides users and OEMs a quick way to get back to square-one, if something went awry or if a user wants to start over. • $RST = # : Erases and zeros all G54-G59 work coordinate offsets and G28/30 positions stored in EEPROM. These are generally the values seen in the $# parameters printout.

dans les paramètres impression # $. Cela fournit un moyen facile d'effacer sans avoir à le faire manuellement pour chaque ensemble avec une commande G20 L2 / 20 ou G28.1 / 30. 1	This provides an easy way to clear these without having to do it manually for each set with a G20 L2/20 or G28.1/30.1 command.
• $RST = * : Ceci supprime et restaure toutes les données de l'EEPROM utilisés par Grbl. Cela inclut les paramètres $$, $#, les lignes de démarrage $N et la chaîne d'information $I. Notez que cela ne nettoie pas toute l' EEPROM, seules les zones de données utilisées par Grbl. Pour faire un nettoyage complet, s'il vous plaît utiliser le projet nettoyage de l'EEPROM Arduino.	• $RST =* : This clears and restores all of the EEPROM data used by Grbl. This includes $$ settings, $# parameters, $N startup lines, and $I build info string. Note that this doesn't wipe the entire EEPROM, only the data areas Grbl uses. To do a complete wipe, please use the Arduino IDE's EEPROM clear example project.
Real-Time Commands: ~, !, ?, and Ctrl-X	
Les quatre dernières commandes de Grbl sont des commandes en temps réel. Cela signifie qu'elles peuvent être envoyées à tout moment, n'importe où et Grbl répondra immédiatement, peu importe ce qu'il fait. Pour ceux qui sont curieux, ce sont des caractères spéciaux qui sont sélectionnés à partir du flux de série entrant et diront à Grbl de les exécuter, habituellement en quelques 5 millisecondes.	The last four of Grbl's commands are real-time commands. This means that they can be sent at anytime, anywhere, and Grbl will immediately respond, no matter what it's doing. For those that are curious, these are special characters that are 'picked-off' from the incoming serial stream and will tell Grbl to execute them, usually within a few milliseconds.
~ - Cycle start Ceci est le cycle démarrer ou reprendre la commande qui peut être émis à tout moment car c'est une commande en temps réel. Lorsque Grbl a des mouvements en attente dans sa mémoire tampon et est prêt à fonctionner, la commande ~ début de cycle va commencer l'exécution de la mémoire tampon et Grbl commencera à déplacer les axes. Toutefois, par défaut, l'auto cycle est activé, de nouveaux utilisateurs n'auront pas besoin de cette commande à moins qu'un redémarrage ne soit effectué. Quand un redémarrage est exécuté, le départ cycle reprendra le programme. Le démarrage du cycle ne sera efficace que quand il y a des ordres dans le tampon prêts à s'exécuter et ne fonctionnera pas avec d'autres processus comme homing.	**~ - Cycle start** This is the cycle start or resume command that can be issued at any time, as it is a real-time command. When Grbl has motions queued in its buffer and is ready to go, the ~ cycle start command will start executing the buffer and Grbl will begin moving the axes. However, by default, auto-cycle start is enabled, so new users will not need this command unless a feed hold is performed. When a feed hold is executed, cycle start will resume the program. Cycle start will only be effective when there are motions in the buffer ready to go and will not work with any other process like homing.
! - Feed hold La commande d'alimentation de maintien apportera le cycle actif à un arrêt via une décélération contrôlée, afin de ne pas perdre la position. Elle est également en temps réel et peut être activée à tout moment. Une fois terminé ou suspendu, Grbl attendra une commande de cycle de démarrage pour reprendre le programme. Faire une pause ne peut interrompre qu'un cycle et n'affectera pas homing ou tout autre processus. Si vous avez besoin d'arrêter un cycle à mi-programme et ne pas perdre la position, effectuer une prise d'alimentation à Grbl comme pour un arrêt contrôlé. Une fois terminé, vous pouvez effectuer une réinitialisation. Toujours essayer d'exécuter une prise d'alimentation lorsque la machine est en marche avant de faire une réinitialisation, sauf bien sûr s'il y a une situation d'urgence.	**! - Feed hold** The feed hold command will bring the active cycle to a stop via a controlled deceleration, so as not to lose position. It is also real- time and may be activated at any time. Once finished or paused, Grbl will wait until a cycle start command is issued to resume the program. Feed hold can only pause a cycle and will not affect homing or any other process. If you need to stop a cycle mid-program and can't afford losing position, perform a feed hold to have Grbl bring everything to a controlled stop. Once finished, you can then issue a reset. Always try to execute a feed hold whenever the machine is running before hitting reset, except of course if there is some emergency situation.
? - Current status La commande ? retrouve immédiatement l'état actif de Grbl et la position actuelle en temps réel, à la fois dans les coordonnées de la machine et les coordonnées de travail. En option, Grbl pourra aussi répondre en retour avec l'utilisation de la mémoire tampon série RX et planificateur via le réglage de masque de rapport de situation. La commande ? peut être envoyée à tout moment et fonctionne de manière asynchrone avec tous les autres processus que Grbl fait. Le réglage Grbl $13 détermine les valeurs en millimètres ou en pouces. Quand ? est pressé, Grbl répond immédiatement avec quelque chose comme ce qui suit : Les états actifs Grbl peuvent être dans : **Idle, Run, Hold, Door, Home, Alarme, Check**	**? - Current status** The ? command immediately returns Grbl's active state and the real-time current position, both in machine coordinates and work coordinates. Optionally, you can also have Grbl respond back with the RX serial buffer and planner buffer usage via the status report mask setting. The ? command may be sent at any time and works asynchronously with all other processes that Grbl is doing. The $13 Grbl setting determines whether it reports millimeters or inches. When ? is pressed, Grbl will immediately reply with something like the following : The active states Grbl can be in are: **Idle, Run, Hold, Door, Home, Alarm, Check**

• **Idle :** Tous les systèmes sont prêts, aucun déplacement en file d'attente et est prêt à toute commande. • **Run :** Indique qu'un cycle est en marche. • **Hold :** attente d'alimentation en cours d'exécution ou ralentissement pour un arrêt. Après l'arrêt complet, Grbl restera en attente d'un début de cycle pour reprendre le programme. • **Door :** (Nouveau en v 0.9i) Cette compilation d'options entraîne Grbl pour nourrir attente, l'arrêt de la broche et du liquide de refroidissement et attend que l'interrupteur de la porte a été fermé et l'utilisateur a émis un début de cycle. Utile pour les OEM qui ont besoin de portes de sécurité. • **Home :** Au milieu d'un cycle de homing. NOTE: Les positions ne sont pas mises à jour en direct pendant le cycle de homing d'origine mais elles vont être mises à la position du point zéro de référence une fois cette action faite. • **Alarm :** Cela indique que quelque chose a mal tourné ou Grbl ne connaît pas sa position. Cet état verrouille toutes les commandes G-code mais vous permet d'interagir avec les paramètres de Grbl si vous en avez besoin. '$X' Alarme déverrouille le blocage et met Grbl en état de veille, ce qui vous permettra de poursuivre votre travail. Comme dit précédemment, se méfier de ce que vous faites après une alarme. • **Check :** Grbl est en mode contrôle G-code. Il permettra de traiter et répondre à toutes les commandes G-code mais pas le mouvement ou d'allumer quoi que ce soit. Une fois basculée avec une autre commande '$C', Grbl se réinitialisera	• **Idle**: All systems are go, no motions queued, and it's ready for anything. • **Run**: Indicates a cycle is running. • **Hold**: A feed hold is in process of executing, or slowing down to a stop. After the hold is complete, Grbl will remain in Hold and wait for a cycle start to resume the program. • **Door**: (New in v0.9i) This compile-option causes Grbl to feed hold, shut-down the spindle and coolant, and wait until the door switch has been closed and the user has issued a cycle start. Useful for OEM that need safety doors. • **Home**: In the middle of a homing cycle. NOTE: Positions are not updated live during the homing cycle, but they'll be set to the home position once done. • **Alarm**: This indicates something has gone wrong or Grbl doesn't know its position. This state locks out all G- code commands, but allows you to interact with Grbl's settings if you need to. '$X' kill alarm lock releases this state and puts Grbl in the Idle state, which will let you move things again. As said before, be cautious of what you are doing after an alarm. • **Check**: Grbl is in check G-code mode. It will process and respond to all G-code commands, but not motion or turn on anything. Once toggled off with another '$C' command, Grbl will reset itself.
`Ctrl-x` - Reset Grbl Cette commande est la réinitialisation logicielle de Grbl. Elle est en temps réel et peut être envoyée à tout moment. Comme son nom l'indique, il réinitialise Grbl mais d'une manière contrôlée, conserve la position de la machine et tout se fait sans éteindre votre Arduino. Les seules fois où un soft-reset pourrait perdre la position est lorsque des problèmes surgissent et des défauts moteurs alors qu'ils se déplaçaient. Si oui, il va rapporter si le suivi de Grbl de position de la machine a été perdu. En effet, une décélération incontrôlée peut conduire à des pas perdus et Grbl ne pourra signaler combien il en a perdu (ce qui est le problème avec les pas à pas en général). Sinon, Grbl sera tout simplement ré-initialisé, exécutera les lignes de démarrage et continuera son petit bonhomme de chemin. S'il vous plaît notez qu'il est recommandé de faire un soft-reset avant de commencer un travail. Cela garantit qu'il n'y a pas de modes actifs ou la reconfiguration de votre machine avant d'exécuter la tâche G-code. Donc, votre machine partira toujours sur des bases saines et fera ce que vous attendez d'elle.	**`Ctrl-x` - Reset Grbl** This is Grbl's *soft* reset command. It's real-time and can be sent at any time. As the name implies, it resets Grbl, but in a controlled way, *retains* your machine position, and all is done without powering down your Arduino. The only times a soft-reset could lose position is when problems arise and the steppers were killed while they were moving. If so, it will report if Grbl's tracking of the machine position has been lost. This is because an uncontrolled deceleration can lead to lost steps, and Grbl has no feedback to how much it lost (this is the problem with steppers in general). Otherwise, Grbl will just re-initialize, run the startup lines, and continue on its merry way. Please note that it's recommended to do a soft-reset before starting a job. This guarantees that there aren't any G-code modes active that from playing around or setting up your machine before running the job. So, your machine will always starts fresh and consistently, and your machine does what you expect it to.