#### Quick-Links:

* [Grbl $ System Commands](https://github.com/gnea/grbl/wiki/Grbl-v1.1-Commands#grbl--commands) Grbl $系统命令
* [Grbl Real-Time Commands](https://github.com/gnea/grbl/wiki/Grbl-v1.1-Commands#grbl-v11-realtime-commands) Grbl实时命令

### More up-to-date documentation is in the /doc/markdown/ directory of the source code. This page is for quick reference. If you spot a problem, feel free to update it or notify us.

### 源代码的/ doc / markdown /目录中有更新的文档。 此页面供快速参考。 如果您发现问题，请随时更新或通知我们。

**Grbl v1.1 Commands**

**Grbl v1.1命令**

In general, Grbl assumes all characters and streaming data sent to it is g-code and will parse and try to execute it as soon as it can. However, Grbl also has two separate system command types that are outside of the normal g-code streaming. One system command type is streamed to Grbl like g-code, but starts with a $ character to tell Grbl it's not g-code. The other is composed of a special set of characters that will immediately command Grbl to do a task in real-time. It's not part of the g-code stream. Grbl's system commands do things like control machine state, report saved parameters or what Grbl is doing, save or print machine settings, run a homing cycle, or make the machine move faster or slower than programmed. This document describes these "internal" system Grbl commands, what they do, how they work, and how to use them.

通常，Grbl假定发送给它的所有字符和流数据都是g代码，并且将解析并尝试尽快执行它。 但是，Grbl还具有两种单独的系统命令类型，它们不在常规g代码流传输之外。 一种系统命令类型像g代码一样流式传输到Grbl，但是以$字符开头告诉Grbl不是g代码。 另一个由一组特殊的字符组成，这些字符将立即命令Grbl实时执行任务。 它不是g代码流的一部分。 Grbl的系统命令可以执行诸如控制机器状态，报告已保存的参数或Grbl正在执行的操作，保存或打印机器设置，运行归位周期或使机器的运行快于或慢于编程等功能。 本文档介绍了这些“内部”系统Grbl命令，它们的作用，如何工作以及如何使用它们。

**Getting Started**

**入门**

First, connect to Grbl using the serial terminal of your choice.

首先，使用您选择的串行终端连接到Grbl。

Set the baud rate to **115200** as 8-N-1 (8-bits, no parity, and 1-stop bit.)

将波特率设置为8-N-1（8位，无奇偶校验和1位停止位）为115200。

Once connected you should get the Grbl-prompt, which looks like this:

一旦连接，你应该得到如下Grbl提示：

Grbl 1.1e ['$' for help]

Type $ and press enter to have Grbl print a help message. You should not see any local echo of the $ and enter. Grbl should respond with:

键入$并按Enter键使Grbl打印一条帮助消息。 你不应该看到$的任何本地回复。 Grbl应该回应：

[HLP:$$ $# $G $I $N $x=val $Nx=line $J=line $SLP $C $X $H ~ ! ? ctrl-x]

The ‘$’-commands are Grbl system commands used to tweak the settings, view or change Grbl's states and running modes, and start a homing cycle. The last four non-'$' commands are realtime control commands that can be sent at anytime, no matter what Grbl is doing. These either immediately change Grbl's running behavior or immediately print a report of the important realtime data like current position (aka DRO). There are over a dozen more realtime control commands, but they are not user type-able. See realtime command section for details.

'$'命令是用于调整设置，查看或更改Grbl状态和运行模式的Grbl系统命令，并启动归位循环。最后四个非“$”命令是可随时发送的实时控制命令，无论Grbl正在做什么。这些都会立即改变Grbl的运行行为，或者立即打印一份重要实时数据的报告，如当前位置（也称为DRO）。有超过十几个实时控制命令，但它们不是用户类型的。 有关详细信息，请参阅实时命令部分。

**Grbl '$' Commands**

**Grbl'$'命令**

The $ system 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.

$ system命令为用户提供额外的控件，例如打印关于当前G代码解析器模态的反馈或运行归位循环。本节介绍这些命令以及如何使用它们。

**$$and $x=val - View and write Grbl settings 查看和写入Grbl设置**

See [Grbl v1.1 Configuration](https://github.com/gnea/grbl/wiki/Grbl-v1.1-Configuration#grbl-settings) for more details on how to view and write setting and learn what they are.

有关如何查看和写入设置以及了解其内容的详细信息，请参阅Grbl v1.1配置。

**$# - View gcode parameters 查看gcode参数**

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.

G代码参数存储G54-G59工作坐标，G28 / G30预定义位置，G92坐标偏移，刀具长度偏移和探测的坐标偏移值（不是正式的，但我们在这里添加）。大多数这些参数直接写入EEPROM，只要它们被更改并持续存在。 这意味着它们将保持不变，不管断电，直到它们被明确改变为止。在复位或上电循环时不保留的非持久参数为G92，G43.1刀具长度偏移和G38.2探测数据。

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.

G54-G59工作坐标可以通过NIST gcode标准和EMC2（linuxcnc.org）标准定义的G10 L2 Px或G10 L20 Px命令进行更改。 G28 / G30预定义位置可以分别通过G28.1和G30.1命令进行更改。

When $# is called, Grbl will respond with the stored offsets from machine coordinates for each system as follows. TLO denotes tool length offset (for the default z-axis), and PRB denotes the coordinates of the last probing cycle, where the suffix :1 denotes if the last probe was successful and :0 as not successful.

当$＃被调用时，Grbl将按照以下方式对每个系统的机器坐标中存储的偏移进行响应。 TLO表示刀具长度偏移（用于默认z轴），PRB表示最后一个探测周期的坐标，其中后缀：1表示如果最后一个探针成功，并且：0不成功。

[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]

[PRB:0.000,0.000,0.000:0]

**$G - View gcode parser state 查看gcode解析器状态**

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 a message starting with an [GC: indicator like:

此命令打印Grbl的G代码解析器中的所有活动的gcode模式。当向Grbl发送此命令时，它将以[GC：指示符开头的消息开始回复：

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

这些主动模式决定了Grbl的G代码解析器将如何解释下一个G代码块或命令。对于那些新的G代码和CNC加工，模式将解析器设置为特定状态，因此您不必不断地告诉解析器如何解析它。这些模式被组织成称为“模态组”的集合，它们不能同时在逻辑上处于活动状态。例如，单位模态组设置您的G代码程序是以英寸还是以毫米为单位。

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](http://www.linuxcnc.org/docs/2.4/html/gcode_overview.html#sec:Modal-Groups). The G-code commands in **bold** indicate the default modes upon powering-up Grbl or resetting it.

Grbl支持的模式组的简短列表如下所示，但在LinuxCNC的网站上可以找到更完整和详细的描述。 G代码粗体字命令表示启动Grbl或重置时的默认模式。

| **Modal Group Meaning**  **模组意义** | **Member Words 成员词** |
| --- | --- |
| Motion Mode  运动模式 | **G0**, G1, G2, G3, G38.2, G38.3, G38.4, G38.5, G80 |
| Coordinate System Select  坐标系选择 | **G54**, G55, G56, G57, G58, G59 |
| Plane Select  平面选择 | **G17**, G18, G19 |
| Distance Mode  距离模式 | **G90**, G91 |
| Arc IJK Distance Mode  圆弧IJK 距离模式 | **G91.1** |
| Feed Rate Mode  进给速率模式 | G93, **G94** |
| Units Mode  单位模式 | G20, **G21** |
| Cutter Radius Compensation  刀具半径补偿 | **G40** |
| Tool Length Offset  工具长度偏移 | G43.1, **G49** |
| Program Mode  程序模式 | **M0**, M1, M2, M30 |
| Spindle State  主轴状态 | M3, M4, **M5** |
| Coolant State  冷却状态 | 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.

除了G代码解析器模式，Grbl将报告有效的T刀具编号，S主轴转速和F进给速率，复位时全部默认为0。对于那些好奇的人来说，这些不太适合使用不错的模式组，但对确定解析器状态来说同样重要。

**$I - View build info 查看构建信息**

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. This string will be saved as capitalized, white space removed, and can only contain alpha-numeric characters. The next time you query Grbl with a $I view build info, Grbl will print this string after the version and build date.

这将向用户打印反馈给Grbl版本和源代码生成日期。或者，如果您使用Grbl的机器多，我也可以存储一个短字符串来帮助识别您正在与哪个CNC机器进行通信。 要设置此字符串，请发送Grbl $ I = xxx，其中xxx是您的自定义字符串，小于80个字符。 这个字符串将被保存为大小写，删除空格，只能包含字母数字字符。下一次使用$ I查看构建信息查询Grbl时，Grbl将在版本和构建日期之后打印此字符串。

NOTE: Some OEMs may block access to over-writing the build info string so they can store product information and codes there.

注意：某些OEM可能会阻止访问重写构建信息字符串，以便它们可以存储产品信息和代码。

**$N - View startup blocks 查看启动块**

$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.

$ Nx是Grbl每次启动Grbl或重置Grbl时运行的启动块。换句话说，一个启动块是一行G代码，您可以让Grbl自动运行，以设置您的G代码模态默认值，或者您每次启动机器时都需要Grbl。 Grbl可以存储两个G代码块作为系统默认值。

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

所以，当连接到Grbl时，键入$ N然后输入。 Grbl应该像以下一样短暂的回应：

$N0=

$N1=

ok

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.

没有太多的事情要做，但这只是意味着在启动时，Grbl的行$ N0中没有存储G代码块。 $ N1是要运行的下一行。

$Nx=line - Save startup block 保存启动块

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.

重要提示：在启动块中存储任何运动（G0 / 1，G2 / 3，G28 / 30）命令时要非常小心。这些运动命令将在您每次重置或启动Grbl时运行，因此如果您有紧急情况，并且必须停止并重置，启动块移动可能会使事情变得更快。另外，不要将任何将数据保存到EEPROM的命令，如G10 / G28.1 / G30.1。这将导致Grbl在每次启动和重置时不断重新写入数据，最终将磨损您的Arduino的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.**

启动块的典型用法是简单地设置您的首选模式状态，例如G20英寸模式，始终默认为不同的工作坐标系，或者为用户提供一种用于运行他们需要的用户编写的独特功能的方式为他们疯狂的项目。

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.

要设置启动块，请键入$ N0 =，后跟一个有效的G代码块和一个输入。 Grbl将运行该块以检查它是否有效，然后以ok或错误回复：告诉您是否成功或出现问题。 如果有错误，Grbl将不会保存。

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.

例如，假设要使用您的第一个启动块$ N来设置G54代码解析器模式，如G54工作坐标，G20英寸模式，G17 XY平面。 您可以输入$ N0 = G20 G54 G17，您应该看到ok响应。 然后，您可以通过键入$ N来检查是否存储，您现在应该看到像$ N0 = G20G54G17这样的响应。

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, starting with an open-chevron >, and a :ok response from Grbl to indicate if it ran okay. So for the previous example, you'll see:

一旦您有一个存储在Grbl的EEPROM中的启动块，每次启动或重新启动时，您将看到您的启动块打印回您，从开放式雪佛龙开始，以及来自Grbl的：ok响应来指示是否运行正常。所以对于前面的例子，你会看到：

Grbl 1.1d ['$' for help]

>G20G54G17:ok

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.

如果您有多个G代码启动块，则每次启动时都会按顺序打印。 如果要清除其中一个启动块（例如，块0），则键入$ N0 =没有任何跟着等号的东西。

NOTE: There are two variations on when startup blocks with run. First, it will not run if Grbl initializes up in an ALARM state or exits an ALARM state via an $X unlock for safety reasons. Always address and cancel the ALARM and then finish by a reset, where the startup blocks will run at initialization. Second, if you have homing enabled, the startup blocks will execute immediately after a successful homing cycle, not at startup.

注意：运行启动块时有两种变体。 首先，如果Grbl在ALARM状态下初始化，或者出于安全原因通过$ X解锁退出ALARM状态，则不会运行。 始终寻址并取消报警，然后通过重置完成，启动块将在初始化时运行。其次，如果启用归位，启动块将在成功的归位循环后立即执行，而不是启动。

**$C - Check gcode mode 检查gcode模式**

This toggles the Grbl's gcode parser to take all incoming blocks and 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).

这将切换Grbl的gcode解析器以获取所有传入的块并完全处理它们，就像在正常操作中一样，但它不会移动任何轴，忽略住宅并关闭主轴和冷却液。这是为了向用户提供一种方法来检查他们的新G代码程序如何与Grbl的解析器一起收费，并监视任何错误（如果启用了软限制违规则检查）。

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.

当切换时，Grbl将执行自动软复位（^ X）。 这是为了两个目的。 它简化了代码管理。但是，当G代码模式不是他们认为的时候，它也阻止用户开始工作。 系统复位始终为用户提供一个新的，一致的开始。

**$X - Kill alarm lock 关闭报警锁**

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.

Grbl的报警模式是某种事情发生严重错误的状态，例如一个周期内的硬限制或中止，或者Grbl不知道其位置。默认情况下，如果您启用并启动了Arduino，Grbl将进入报警状态，因为它不知道其位置。 报警模式将锁定所有G代码命令，直到执行“$ H”归位循环。 或者如果用户需要覆盖报警锁定以将其移动到其限位开关上，例如，“$ X”停止报警锁将覆盖锁定，并允许G代码功能再次工作。

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.

但是，仔细踩踏！ 这只能在紧急情况下使用。 这个职位可能已经失去了，Grbl可能不是你认为的地方。因此，建议使用G91增量模式进行短暂动作。 然后，执行回原点循环或立即重置。

As noted earlier, startup lines do not execute after a $X command. Always reset when you have cleared the alarm and fixed the scenario that caused it. When Grbl resets to idle, the startup lines will then run as normal.

如前所述，启动行在$ X命令之后不执行。 当您清除报警并修复导致报警的场景时，请始终重置。当Grbl复位到空闲时，启动行将正常运行。

**$H - Run homing cycle 运行归位循环**

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.

此命令是在Grbl中执行归位循环的唯一方法。一些其他运动控制器指定一个特殊的G代码命令来运行回原点循环，但根据G代码标准，这是不正确的。归位是由控制器处理的完全独立的命令。

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.

提示：运行归位循环后，您可以手动将所有时间逐步移动到工作空间容积中间的位置。您可以将G28或G30预定义位置设置为您的后归位置，更接近您将要加工的位置。要设置这些，您首先需要将您的机器慢跑到要归零之后要移动的位置。 类型G28.1（或G30.1）使Grbl存储该位置。 那么在'$ H'归位之后，你可以进入'G28'（或'G30'），它会自动地移动到这里。 一般来说，我只是将XY轴移动到中心并离开Z轴。这样可确保主轴中的刀具不会发生干扰，并且不会碰到任何东西。

**$Jx=line - Run jogging motion 运行慢跑运动**

New to Grbl v1.1, this command will execute a special jogging motion. There are three main differences between a jogging motion and a motion commanded by a g-code line.

新到Grbl v1.1，此命令将执行特殊的慢跑运动。 慢跑运动和g代码行指令运动有三个主要区别。

* Like normal g-code commands, several jog motions may be queued into the planner buffer, but the jogging can be easily canceled by a jog-cancel or feed-hold real-time command. Grbl will immediately hold the current jog and then automatically purge the buffers of any remaining commands.
* 像正常的g代码命令一样，可以将几个点动运动排队到计划器缓冲区中，但可以通过点动取消或进给保持实时命令轻松取消点动。 Grbl将立即保持当前点动，然后自动清除任何剩余命令的缓冲区。
* Jog commands are completely independent of the g-code parser state. It will not change any modes like G91 incremental distance mode. So, you no longer have to make sure that you change it back to G90 absolute distance mode afterwards. This helps reduce the chance of starting with the wrong g-code modes enabled.
* 点动命令完全独立于g代码解析器状态。 它不会改变任何模式，如G91增量距离模式。 所以，您不再需要确保将其更改回G90绝对距离模式。这有助于减少启用错误的g代码模式的机会。
* If soft-limits are enabled, any jog command that exceeds a soft-limit will simply return an error. It will not throw an alarm as it would with a normal g-code command. This allows for a much more enjoyable and fluid GUI or joystick interaction.
* 如果启用了软限制，任何超过软限制的点动命令都将简单地返回错误。 它不会像正常的g代码命令一样发出警报。这允许更加愉快和流畅的GUI或操纵杆交互。

Executing a jog requires a specific command structure, as described below:

执行慢跑需要一个特定的命令结构，如下所述：

* The first three characters must be '$J=' to indicate the jog.
* 前三个字符必须为'$ J ='以指示点动。
* The jog command follows immediate after the '=' and works like a normal G1 command.
* 点动命令在'='之后立即执行，并且像普通G1命令一样工作。
* Feed rate is only interpreted in G94 units per minute. A prior G93 state is ignored during jog.
* 进给速率仅以G94单位/分钟解释。 在点动期间，先前的G93状态被忽略。
* Required words:
* 所需词语：
  + XYZ: One or more axis words with target value.
  + XYZ：具有目标值的一个或多个轴字。
  + F - Feed rate value. NOTE: Each jog requires this value and is not treated as modal.
  + F - 进给速率值。注意：每次点动都需要该值，不能视为模态。
* Optional words: Jog executes based on current G20/G21 and G90/G91 g-code parser state. If one of the following optional words is passed, that state is overridden for one command only.
* 可选字：基于当前G20 / G21和G90 / G91 g代码解析器状态执行点动。如果传递了以下可选字之一，那么该状态仅适用于一个命令。
  + G20 or G21 - Inch and millimeter mode
  + G20或G21 - 英制和毫米模式
  + G90 or G91 - Absolute and incremental distances
  + G90或G91 - 绝对和增量距离
  + G53 - Move in machine coordinates
  + G53 - 移动机床坐标
* All other g-codes, m-codes, and value words are not accepted in the jog command.
* 所有其他g代码，m代码和值字在jog命令中不被接受。
* Spaces and comments are allowed in the command. These are removed by the pre-parser.
* 命令中允许使用空格和注释。 这些由预解析器删除。
* Example: G21 and G90 are active modal states prior to jogging. These are sequential commands.
* 示例：G21和G90是慢跑前的活动状态。 这些是顺序命令。
  + $J=X10.0 Y-1.5 will move to X=10.0mm and Y=-1.5mm in work coordinate frame (WPos).
  + $ J = X10.0在工作坐标系（WPos）中，Y-1.5将移动到X = 10.0mm，Y = -1.5mm。
  + $J=G91 G20 X0.5 will move +0.5 inches (12.7mm) to X=22.7mm (WPos). Note that G91 and G20 are only applied to this jog command.
  + $ J = G91 G20 X0.5将移动+0.5英寸（12.7mm）至X = 22.7mm（WPos）。请注意，G91和G20仅适用于该jog指令。
  + $J=G53 Y5.0 will move the machine to Y=5.0mm in the machine coordinate frame (MPos). If the work coordinate offset for the y-axis is 2.0mm, then Y is 3.0mm in (WPos).
  + $ J = G53 Y5.0将机器移动到机器坐标系（MPos）中的Y = 5.0mm。如果y轴的工作坐标偏移量为2.0mm，则（WPos）中的Y为3.0mm。

Jog commands behave almost identically to normal g-code streaming. Every jog command will return an 'ok' when the jogging motion has been parsed and is setup for execution. If a command is not valid or exceeds a soft-limit, Grbl will return an 'error:'. Multiple jogging commands may be queued in sequence.

点动命令的行为与正常的g代码流几乎相同。 当慢跑运动已解析并设置执行时，每个jog命令都将返回“ok”。如果命令无效或超过软限制，则Grbl将返回“error：”。 多个点动命令可以按顺序排队。

NOTE: See additional jogging documentation for details on using this command to create a low-latency joystick or rotary dial interface.

注意：有关使用此命令创建低延迟操纵杆或旋转拨号接口的详细信息，请参阅其他慢跑文档。

$RST=$, $RST=#, and $RST=\*- Restore Grbl settings and data to defaults

$ RST = $，$ RST =＃，$ RST = \* - 将Grbl设置和数据恢复为默认值

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.

这些命令不在主Grbl $帮助消息中列出，但可用于允许用户恢复Grbl的EEPROM数据的部分或全部。 注意：执行这些命令之一后，Grbl将自动复位，以确保系统正确初始化。

$RST=$ : 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 = $：将$$ Grbl设置擦除并恢复为默认值，该值由编译Grbl时使用的默认设置文件定义。 通常，OEM将通过机器特定的推荐设置构建其Grbl固件。 这为用户和原始设备制造商提供了一个快速的方法来回到平方，如果出现问题，或者用户想要重新启动。

$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. This provides an easy way to clear these without having to do it manually for each set with a G20 L2/20 orG28.1/30.1 command.

$ RST =＃：擦除和保存所有G54-G59工作坐标偏移和存储在EEPROM中的G28 / 30位置。 这些通常是$＃参数打印输出中看到的值。 这提供了一个简单的方法来清除这些，而无需使用G20 L2 / 20或G28.1 / 30.1命令为每个组手动执行。

$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.

$ RST = \*：这将清除并恢复Grbl使用的所有EEPROM数据。 这包括$$设置，$＃参数，$ N启动行和$ I构建信息字符串。 请注意，这不会擦除整个EEPROM，只有Grbl使用的数据区域。 要完整擦拭，请使用Arduino IDE的EEPROM清除示例项目。

NOTE: Some OEMs may restrict some or all of these commands to prevent certain data they use from being wiped.

注意：某些OEM可能会限制部分或全部命令，以防止其使用的某些数据被擦除。

$SLP - Enable Sleep Mode 启用睡眠模式

This command will place Grbl into a de-powered sleep state, shutting down the spindle, coolant, and stepper enable pins and block any commands. It may only be exited by a soft-reset or power-cycle. Once re-initialized, Grbl will automatically enter an ALARM state, because it's not sure where it is due to the steppers being disabled.

该命令将Grbl置于停电状态，关闭主轴，冷却液和步进器使能引脚并阻止任何命令。 它只能通过软复位或电源循环退出。 一旦重新初始化，Grbl将自动进入ALARM状态，因为它不确定由于步进器被禁用而在哪里。

This feature is useful if you need to automatically de-power everything at the end of a job by adding this command at the end of your g-code program, BUT, it is highly recommended that you add commands to first move your machine to a safe parking location prior to this sleep command. It also should be emphasized that you should have a reliable CNC machine that will disable everything when its supposed to, like your spindle. Grbl is not responsible for any damage it may cause. It's never a good idea to leave your machine unattended. So, use this command with the utmost caution!

如果您需要通过在g代码程序的末尾添加此命令来自动关闭作业结束时的所有功能，则此功能非常有用，但是强烈建议您添加命令以首先将机器移动到 在这个睡眠命令之前的安全停车位置。 还应该强调的是，你应该有一个可靠的数控机床，它将禁用所有的东西，当它像你的主轴。 Grbl对任何可能导致的损坏概不负责。 将您的机器无人值守，这绝对不是个好主意。 所以使用这个命令非常小心！

**Grbl v1.1 Realtime commands**

**Grbl v1.1实时命令**

Realtime commands are single control characters that may be sent to Grbl to command and perform an action in real-time. This means that they can be sent at anytime, anywhere, and Grbl will immediately respond, regardless of what it is doing at the time. These commands include a reset, feed hold, resume, status report query, and overrides (in v1.1).

实时命令是单个控制字符，可以发送到Grbl来实时执行操作。 这意味着它们可以随时随地发送，而且Grbl将立即作出回应，无论当时正在做什么。 这些命令包括复位，进给保持，恢复，状态报告查询和覆盖（在v1.1中）。

A realtime command:实时指令：

* Will execute within tens of milliseconds.
* 将在几十毫秒内执行。
* Is a single character that may be sent to Grbl at any time.
* 可以随时发送到Grbl的单个字符。
* Does not require a line feed or carriage return after them.
* 他们不需要换行或回车。
* Is not considered a part of the streaming protocol.
* 不被认为是流媒体协议的一部分。
* Are intercepted when they are received and never placed in a buffer to be parsed by Grbl.
* 被收到时被拦截，并且不被放置在缓冲区中以供Grbl解析。
* Will ignore multiple commands until it has executed the first received command.
* 在执行第一个接收到的命令之前，将忽略多个命令。
* May be tied to an input pin and may be operated with a button or switch.
* 可以连接到输入引脚，并可以用按钮或开关操作。
* Actions depends on state or what Grbl is doing. It may not do anything.
* 行动取决于国家或Grbl正在做什么。 它可能不会做任何事情。
* Descriptions explain how they work and what to expect.
* 说明如何工作和期待什么。

**ASCII Realtime Command Descriptions**

**ASCII实时命令说明**

Four realtime commands are type-able by users on a keyboard and shown in the $ Grbl help message. These realtime command characters control some of Grbl's basic functions.

键盘上的用户可以使用四个实时命令，并显示在$ Grbl帮助消息中。 这些实时命令字符控制了Grbl的一些基本功能。

* 0x18 (ctrl-x) : Soft-Reset 软复位
  + Immediately halts and safely resets Grbl without a power-cycle.
  + 在没有电源循环的情况下立即停止并安全地重置Grbl。
  + Accepts and executes this command at any time.
  + 随时接受并执行此命令。
  + If reset while in motion, Grbl will throw an alarm to indicate position may be lost from the motion halt.
  + 如果在运动中复位，Grbl会发出报警，指示位置可能会从停止运动中丢失。
  + If reset while in not motion, position is retained and re-homing is not required.
  + 如果在不运动时复位，则保持位置并且不需要重新归位。
  + An input pin is available to connect a button or switch.
  + 输入引脚可用于连接按钮或开关。
* ? : Status Report Query 状态报告查询
  + Immediately generates and sends back runtime data with a status report.
  + 立即生成并发回具有状态报告的运行时数据。
  + Accepts and executes this command at any time, except during a homing cycle and when critical alarm (hard/soft limit error) is thrown.
  + 接受并执行此命令，除了在归位循环期间以及当发生严重报警（硬/软限位错误）时。
* ~ : Cycle Start / Resume 循环启动/恢复
  + Resumes a feed hold, a safety door/parking state when the door is closed, and the M0 program pause states.
  + 当门关闭时，恢复进给保持，安全门/停车状态，M0程序暂停状态。
  + Command is otherwise ignored.
  + 否则忽略命令。
  + If the parking compile-time option is enabled and the safety door state is ready to resume, Grbl will re-enable the spindle and coolant, move back into position, and then resume.
  + 如果停车编译时间选项被启用，并且安全门状态准备好恢复，则Grbl将重新启用主轴和冷却液，移回到位，然后恢复。
  + An input pin is available to connect a button or switch.
  + 输入引脚可用于连接按钮或开关。
* ! : Feed Hold 进给保持
  + Places Grbl into a suspend or HOLD state. If in motion, the machine will decelerate to a stop and then be suspended.
  + 将Grbl置于挂起状态或HOLD状态。 如果运动，机器将减速停止，然后暂停。
  + Command executes when Grbl is in an IDLE, RUN, or JOG state. It is otherwise ignored.
  + 当Grbl处于IDLE，RUN或JOG状态时，执行命令。 否则被忽略。
  + If jogging, a feed hold will cancel the jog motion and flush all remaining jog motions in the planner buffer. The state will return from JOG to IDLE or DOOR, if was detected as ajar during the active hold.
  + 如果点动，进给保持将取消点动运动，并清除计划器缓冲区中的所有剩余点动。 如果在主动保持期间被检测到为空，状态将从JOG返回到空闲或门禁。
  + By machine control definition, a feed hold does not disable the spindle or coolant. Only motion.
  + 通过机器控制定义，进给保持不会禁用主轴或冷却液。 只有动议。
  + An input pin is available to connect a button or switch.
  + 输入引脚可用于连接按钮或开关。

**Extended-ASCII Realtime Command Descriptions**

**扩展ASCII实时命令说明**

Grbl v1.1 installed more than a dozen new realtime commands to control feed, rapid, and spindle overrides. To help prevent users from inadvertently altering overrides with a keystroke and allow for more commands later on, all of the new control characters have been moved to the extended ASCII character set. These are not easily type-able on a keyboard, but, depending on the OS, they may be entered using specific keystroke and code. GUI developers will need to be able to send extended ASCII characters, values 128 (0x80) to 255 (0xFF), to Grbl to take advantage of these new features.

Grbl v1.1安装了十几个新的实时命令来控制进给，快速和主轴覆盖。 为了防止用户无意中通过按键更改覆盖值，稍后允许更多命令，所有新的控制字符都已被移动到扩展的ASCII字符集。 这些键盘上不容易键入，但根据操作系统，可以使用特定的按键和代码输入。 GUI开发人员需要能够将扩展ASCII字符（值128（0x80））发送到255（0xFF），以便利用这些新功能。

* 0x84 : Safety Door 安全门
  + Although typically connected to an input pin to detect the opening of a safety door, this command allows a GUI to enact the safety door behavior with this command.
  + 虽然通常连接到输入引脚以检测安全门的打开，但是该命令允许GUI通过该命令来产生安全门的行为。
  + Immediately suspends into a DOOR state and disables the spindle and coolant. If in motion, the machine will decelerate to a stop and then be suspended.
  + 立即停止进入DOOR状态，并禁用主轴和冷却液。 如果运动，机器将减速停止，然后暂停。
  + If executed during homing, Grbl will instead halt motion and throw a homing alarm.
  + 如果在归位期间执行，Grbl将停止运动并抛出一个归位报警。
  + If already in a suspend state or HOLD, the DOOR state supersedes it.
  + 如果已处于挂起状态或HOLD状态，则DOOR状态将取代该状态。
  + If the parking compile-time option is enabled, Grbl will park the spindle to a specified location.
  + 如果启用了停车编译时选项，Grbl会将主轴停放在指定位置。
  + Command executes when Grbl is in an IDLE, HOLD, RUN, HOMING, or JOG state. It is otherwise ignored.
  + 当Grbl处于空闲状态，HOLD，RUN，HOMING或JOG状态时，执行命令。 否则被忽略。
  + If jogging, a safety door will cancel the jog and all queued motions in the planner buffer. When the safety door is closed and resumed, Grbl will return to an IDLE state.
  + 如果慢跑，安全门将取消点动和计划器缓冲区中所有排队的运动。 当安全门关闭并恢复时，Grbl将返回空闲状态。
  + An input pin is available to connect a button or switch, if enabled with a compile-time option.
  + 如果启用了编译时选项，则可以使用输入引脚连接按钮或开关。
  + Some builds of Grbl v0.9 used the @ character for this command, but it was undocumented. Moved to extended-ASCII to prevent accidental commanding.
  + Grbl v0.9的一些构建使用了这个命令的@字符，但没有记录。 移动到扩展ASCII以防止意外命令。
* 0x85 : Jog Cancel 慢跑取消
  + Immediately cancels the current jog state by a feed hold and automatically flushing any remaining jog commands in the buffer.
  + 通过进给保持立即取消当前的点动状态，并自动刷新缓冲器中的任何剩余点动命令。
  + Command is ignored, if not in a JOG state or if jog cancel is already invoked and in-process.
  + 命令被忽略，如果不处于JOG状态，或者点动取消已经被调用并处于进行状态。
  + Grbl will return to the IDLE state or the DOOR state, if the safety door was detected as ajar during the cancel.
  + 如果在取消过程中检测到安全门，则Grbl将返回到空闲状态或DOOR状态。
* Feed Overrides 进给覆盖
  + Immediately alters the feed override value. An active feed motion is altered within tens of milliseconds.
  + 立即改变进纸倍率值。 主动进给运动在几十毫秒内改变。
  + Does not alter rapid rates, which include G0, G28, and G30, or jog motions.
  + 不会改变速率，包括G0，G28和G30，或慢跑运动。
  + Feed override value can not be 10% or greater than 200%.
  + 进给倍率值不能超过10％或大于200％。
  + If feed override value does not change, the command is ignored.
  + 如果进给倍率值不变，则忽略该命令。
  + Feed override range and increments may be changed in config.h.
  + 在config.h中可以更改Feed覆盖范围和增量。
  + The commands are:
    - 命令是：
    - 0x90 : Set 100% of programmed rate.
    - 设定编程速率的100％。
    - 0x91 : Increase 10%
    - 0x91：增加10％
    - 0x92 : Decrease 10%
    - 0x92：减少10％
    - 0x93 : Increase 1%
    - 0x93：增加1％
    - 0x94 : Decrease 1%
    - 0x94：减少1％
* Rapid Overrides 快速覆盖
  + Immediately alters the rapid override value. An active rapid motion is altered within tens of milliseconds.
  + 立即改变快速覆盖值。 主动快速运动在几十毫秒内改变。
  + Only effects rapid motions, which include G0, G28, and G30.
  + 只影响快速运动，包括G0，G28和G30。
  + If rapid override value does not change, the command is ignored.
  + 如果快速重写值不更改，则忽略该命令。
  + Rapid override set values may be changed in config.h.
  + config.h中可以更改快速重写设置值。
  + The commands are:
  + 命令是：
    - 0x95 : Set to 100% full rapid rate.设置为100％全速率。
    - 0x96 : Set to 50% of rapid rate.达到快速率的50％。
    - 0x97 : Set to 25% of rapid rate.达到快速率的25％。
* Spindle Speed Overrides主轴速度覆盖
  + Immediately alters the spindle speed override value. An active spindle speed is altered within tens of milliseconds.
  + 立即改变主轴转速倍率值。 主动主轴转速在几十毫秒内被改变。
  + Override values may be changed at any time, regardless of if the spindle is enabled or disabled.
  + 无论主轴是启用还是禁用，可以随时更改覆盖值。
  + Spindle override value can not be 10% or greater than 200%
  + 主轴倍率值不能超过10％或大于200％
  + If spindle override value does not change, the command is ignored.
  + 如果主轴倍率值不变，则忽略该命令。
  + Spindle override range and increments may be altered in config.h.
  + 主机超驰范围和增量可能会在config.h中更改。
  + The commands are:命令是：
    - 0x99 : Set 100% of programmed spindle speed
    - 0x99：设定主轴转速的100％
    - 0x9A : Increase 10%
    - 0x9A：增加10％
    - 0x9B : Decrease 10%
    - 0x9B：减少10％
    - 0x9C : Increase 1%
    - 0x9C：增加1％
    - 0x9D : Decrease 1%
    - 0x9D：减少1％
* 0x9E : Toggle Spindle Stop切换主轴停止
  + Toggles spindle enable or disable state immediately, but only while in the HOLD state.
  + 立即切换主轴启用或禁用状态，但只能处于HOLD状态。
  + The command is otherwise ignored, especially while in motion. This prevents accidental disabling during a job that can either destroy the part/machine or cause personal injury. Industrial machines handle the spindle stop override similarly.
  + 否则忽略该命令，特别是在运动时。 这可以防止在作业中意外失效，从而可能会破坏零件/机器或造成人身伤害。 工业机械类似地处理主轴停止倍率。
  + When motion restarts via cycle start, the last spindle state will be restored and wait 4.0 seconds (configurable) before resuming the tool path. This ensures the user doesn't forget to turn it back on.
  + 当通过循环启动运动重新启动时，将恢复最后一个主轴状态，并等待4.0秒（可配置），然后再恢复刀具路径。 这确保用户不会忘记重新打开它。
  + While disabled, spindle speed override values may still be altered and will be in effect once the spindle is re-enabled.
  + 禁用时，主轴转速倍率值可能仍然会发生变化，一旦重新启动主轴，其将会生效。
  + If a safety door is opened, the DOOR state will supersede the spindle stop override, where it will manage the spindle re-energizing itself upon closing the door and resuming. The prior spindle stop override state is cleared and reset.
  + 如果安全门打开，DOOR状态将取代主轴停止超驰，在关闭门并重新启动时，它将管理主轴自行重新启动。 先前的主轴停止超驰状态被清除并复位。
* 0xA0 : Toggle Flood Coolant切换洪水冷却液
  + Toggles flood coolant state and output pin until the next toggle or g-code command alters it.
  + 切换洪水冷却液状态和输出引脚，直到下一个切换或g代码命令改变它。
  + May be commanded at any time while in IDLE, RUN, or HOLD states. It is otherwise ignored.
  + 随时可以在空闲状态，运行状态或持续状态下进行指令。 否则被忽略。
  + This override directly changes the coolant modal state in the g-code parser. Grbl will continue to operate normally like it received and executed an M8 or M9 g-code command.
  + 该覆盖直接改变g代码解析器中的冷却剂模态。 Grbl将继续正常运行，就像收到并执行M8或M9 g代码命令一样。
  + When $G g-code parser state is queried, the toggle override change will be reflected by anM8 enabled or disabled with an M9 or not appearing when M7 is present.
  + 当查询$ G g代码解析器状态时，触发覆盖更改将被启用或禁用M9的M8反映，或者当M7存在时不显示。
* 0xA1 : Toggle Mist Coolant 切换雾化冷却液
  + Enabled by ENABLE\_M7 compile-time option. Default is disabled.
  + 由ENABLE\_M7编译时选项启用。 默认为禁用。
  + Toggles mist coolant state and output pin until the next toggle or g-code command alters it.
  + 切换雾化冷却液状态和输出管脚直到下一个切换或g代码命令改变它。
  + May be commanded at any time while in IDLE, RUN, or HOLD states. It is otherwise ignored.
  + 随时可以在空闲状态，运行状态或持续状态下进行指令。 否则被忽略。
  + This override directly changes the coolant modal state in the g-code parser. Grbl will continue to operate normally like it received and executed an M7 or M9 g-code command.
  + 该覆盖直接改变g代码解析器中的冷却剂模态。 Grbl将继续正常运行，就像收到并执行M7或M9 g代码命令一样。
  + When $G g-code parser state is queried, the toggle override change will be reflected by anM7 enabled or disabled with an M9 or not appearing when M8 is present.
  + 当查询$ G g代码解析器状态时，触发覆盖更改将被启用或禁用M9的M7反映，或M8出现时不显示。