Grbl v1.1 - Change Summary

Specific details are available in the other markdown documents.

GUI Interface Tweaks from Grbl v0.9

Grbl v1.1's interface protocol has been tweaked in the attempt to make GUI development cleaner, clearer, and hopefully easier. All messages are designed to be deterministic without needing to know the context of the message. Each can be inferred to a much greater degree than before just by the message type, which are all listed below.

- `ok` / `error:x` : Normal send command and execution response acknowledgement. Used for streaming.
- `< >` : Enclosed chevrons contains status report data.
- `Grbl X.Xx ['$' for help]` : Welcome message indicates initialization.
- `ALARM:x` : Indicates an alarm has been thrown. Grbl is now in an alarm state.
- `$x=val` and `$Nx=line` indicate a settings printout from a `$` and `$N` user query, respectively.
- `[MSG:]` : Indicates a non-queried feedback message.
- `[GC:]` : Indicates a queried `$G` g-code state message.
- `[HLP:]` : Indicates the help message.
- `[G54:]`, `[G55:]`, `[G56:]`, `[G57:]`, `[G58:]`, `[G59:]`, `[G28:]`, `[G30:]`, `[G92:]`, `[TLO:]`, and `[PRB:]` messages indicate the parameter data printout from a `$#` user query.
- `[VER:]` : Indicates build info and string from a `$I` user query.
- `[OPT:]` : Indicates compile-time option info from a `$I` user query.
- `[echo:]` : Indicates an automated line echo from a pre-parsed string prior to g-code parsing. Enabled by config.h option.
- `>G54G20:ok` : The open chevron indicates startup line execution. The `:ok` suffix shows it executed correctly without adding an unmatched `ok` response on a new line.

In addition, all `$x=val` settings, `error:`, and `ALARM:` messages no longer contain human-readable strings, but rather codes that are defined in other documents. The `$` help message is also reduced to just showing the available commands. Doing this saves incredible amounts of flash space. Otherwise, the new overrides features would not have fit.

Other minor changes and bug fixes that may effect GUI parsing include:

- Floating point values printed with zero precision do not show a decimal, or look like an integer. This includes spindle speed RPM and feed rate in mm mode.
- `$G` reports fixed a long time bug with program modal state. It always showed `M0` program pause when running. Now during a normal program run, no program modal state is given until an `M0`, `M2`, or `M30` is active and then the appropriate state will be shown.

On a final note, these interface tweaks came about out of necessity, because more data is being sent back from Grbl, it is capable of doing many more things, and flash space is at a premium. It's not intended to be altered again in the near future, if at all. This is likely the only and last major change to this. If you have any comments or suggestions before Grbl v1.1 goes to master, please do immediately so we can all vet the new alteration before its installed.

Realtime Status Reports Changes from Grbl v0.9

- Intent of changes is to make parsing cleaner, reduce transmitting overhead without effecting overall Grbl performance, and add more feedback data, which includes three new override values and real-time velocity.
- Data fields are separated by `|` pipe delimiters, rather than `,` commas that were used to separate data values. This should help with parsing.
- The ability to mask and add/remove data fields from status reports via the `$10` status report mask setting has been disabled. Only selecting `MPos:` or `WPos:` coordinates is allowed.
  - All available data is always sent to standardize the reports across all GUIs.
  - For unique situations, data fields can be removed by config.h macros, but it is highly recommended to not alter these.
- `MPos:` OR `WPos:` are always included in a report, but not BOTH at the same time.
  - This reduces transmit overhead tremendously by removing upwards to 40 characters.
  - `WCO:0.000,10.000,2.500` A current work coordinate offset is now sent to easily convert between position vectors, where `WPos = MPos - WCO` for each axis.
    - `WCO:` is included immediately whenever a `WCO:` value changes or intermittently after every **X** status reports as a refresh. Refresh rates can dynamically vary from 10 to 30 (configurable) reports depending on what Grbl is doing.
    - `WCO:` is simply the sum of the work coordinate system, G92, and G43.1 tool length offsets.
    - Basically, a GUI just needs to retain the last `WCO:` and apply the equation to get the other position vector.
    - `WCO:` messages may only be disabled via a config.h compile-option, if a GUI wants to handle the work position calculations on its own to free up more transmit bandwidth.
  - Be aware of the following issue regarding `WPos:`.
    - In Grbl v0.9 and prior, there is an old outstanding bug where the `WPos:` work position reported may not correlate to what is executing, because `WPos:` is based on the g-code parser state, which can be several motions behind. Grbl v1.1 now forces the planner buffer to empty, sync, and stops motion whenever there is a command that alters the work coordinate offsets `G10,G43.1,G92,G54-59`. This is the simplest way to ensure `WPos:` is always correct. Fortunately, it's exceedingly rare that any of these commands are used need continuous motions through them.
    - A compile-time option is available to disable the planner sync and forced stop, but, if used, it's up to the GUI to handle this position correlation issue.
- The `Hold` and `Door` states includes useful sub-state info via a `:` colon delimiter and an integer value. See descriptions for details.
- Limit and other input pin reports have significantly changed to reduce transmit overhead.
  - The data type description is now just `Pn:`, rather than `Lim:000` or `Pin:000|0|0000`
  - It does not appear if no inputs are detected as triggered.
  - If an input is triggered, `Pn:` will be followed by a letter or set of letters of every triggered input pin. `XYZPDHRS` for the XYZ-axes limits, Probe, Door, Hold, soft-Reset, cycle Start pins, respectively.
  - For example, a triggered Z-limit and probe pin would report `Pn:ZP`.
- Buffer data (planner and serial RX) reports have been tweaked and combined.
  - `Bf:15,128`. The first value is the available blocks in the planner buffer and the second is available bytes in the serial RX buffer.
  - Note that this is different than before, where it reported blocks/bytes "in-use", rather than "available". This change does not require a GUI to know how many blocks/bytes Grbl has been compiled with, which can be substantially different on a Grbl-Mega build.
- Override reports are intermittent since they don't change often once set.
  - Overrides are included in every 10 or 20 status reports (configurable) depending on what Grbl is doing or, if an override value or toggle state changes, automatically in the next report.
  - There are two override fields:
    - `Ov:100,100,100` Organized as feed, rapid, and spindle speed overrides in percent.
- Accessory states are shown alongside override reports when they are active. Like pin states, an accessory state report `A:SFM` contains a letter indicating an active accessory. Letters `S`, `C`, `F`, and `M` are defined as spindle CW, spindle CCW, flood coolant, and mist coolant, respectively. The pins are directly polled and shown here.
- Line numbers, when enabled in config.h, are omitted when:
  - No line number is passed to Grbl in a block.
  - Grbl is performing a system motion like homing, jogging, or parking.
  - Grbl is executing g-code block that does not contain a motion, like `G20G54` or `G4P1` dwell. (NOTE: Looking to fixing this later.)

New Commands

- `$SLP` - Grbl v1.1 now has a sleep mode that can be invoked by this command. It requires Grbl to be in either an IDLE or ALARM state. Once invoked, Grbl will de-energize all connected systems, including the spindle, coolant, and stepper drivers. It'll enter a suspend state that can only be exited by a reset. When reset, Grbl will re-initiatize in an ALARM state because the steppers were disabled and position can not be guaranteed.
  - NOTE: Grbl-Mega can invoke the sleep mode at any time, when the sleep timeout feature is enabled in config.h. It does so when Grbl has not received any external input after a timeout period.
- `$J=line` New jogging commands. This command behaves much like a normal G1 command, but there are some key differences. Jog commands don't alter the g-code parser state, meaning a GUI doesn't have to manage it anymore. Jog commands may be queued and cancelled at any time, where they are automatically flushed from the planner buffer without requiring a reset. See the jogging documentation on how they work and how they may be used to implement a low-latency joystick or rotary dial.
- Laser mode `$` setting - When enabled, laser mode will move through consecutive G1, G2, and G3 motion commands that have different spindle speed values without stopping. A spindle speed of zero will disable the laser without stopping as well. However, when spindle states change, like M3 or M5, stops are still enforced.
  - NOTE: Parking motions are automatically disabled when laser mode is enabled to prevent burning.
- `G56 P1` and `G56 P0` - When enabled in config.h with Grbl's parking motion, these commands enable and disable, respectively, the parking motion. Like all override control commands, these commands are modal and are part of the g-code stream.