PicoMite Stepper Motor Control Reference

1. Overview

The STEPPER command provides a comprehensive system for controlling up to 3 stepper motor axes (X, Y, Z) with support for G-code execution, acceleration planning, and hardware limit switches. It uses a dedicated 100kHz interrupt timer for smooth pulse generation.

2. Initialization & Configuration

2.1 Initialization

```
STEPPER INIT [arc_tolerance] [,buffer_size] [,estop_pin] [,estop_keep_enabled]
```

Initializes the stepper subsystem. Must be called before any other STEPPER commands.
- arc_tolerance: (Optional) Tolerance for arc segmentation in mm (default: 0.05).
- buffer_size: (Optional) Size of the G-code lookahead buffer (default: 32, max: 1024).
- estop_pin: (Optional) Hardware emergency-stop input pin (active low).
- estop_keep_enabled: (Optional) 0 (default) = disable drivers on E-STOP, 1 = keep drivers enabled on E-STOP.

2.2 Axis Configuration

```
STEPPER AXIS axis, step_pin, dir_pin [, enable_pin] [, dir_invert] [, steps_per_mm] [, max_vel] [, max_accel] [, home_backoff_mm]
```

Configures a specific axis (X, Y, or Z).
- axis: X, Y, or Z.
- step_pin/dir_pin: GPIO pins for Step and Direction signals.
- enable_pin: (Optional) GPIO pin for Enable signal (active low).
- dir_invert: (Optional) 1 to invert direction, 0 otherwise.
- steps_per_mm: (Optional) Steps required to move 1mm.
- max_vel: (Optional) Maximum velocity in mm/min.
- max_accel: (Optional) Maximum acceleration in mm/s^2.
- home_backoff_mm: (Optional) Homing switch clear/backoff distance in mm (default: 3.0).

2.3 Limits & Safety

```
STEPPER HWLIMITS x_min, y_min, z_min [,x_max] [,y_max] [,z_max]
```

Configures hardware limit switch pins (active low). Pins must be mutually exclusive with AXIS/SPINDLE/E-STOP, except min and max may share the same pin on the same axis.

```
STEPPER LIMITS axis, min_mm, max_mm
```

Sets soft limits for an axis in mm.

```
STEPPER ESTOP
```

Software emergency stop: halts motion immediately, clears buffer, and turns spindle off. Drivers are disabled unless estop_keep_enabled was set to 1 in STEPPER INIT.
If an INIT estop_pin is configured, hardware E-STOP is monitored in ISR and terminates processing

immediately under all circumstances (including homing).

3. Motion Control

3.1 G-Code Execution

```
STEPPER GC <gcode> [words...]
```

Executes a standard G-code command string.
Supported codes: G0, G1, G2, G3, G4, G28, G90, G91, G92, M3, M5.
M3/M5 and G4 are buffered and executed in-order with motion blocks.
G28 homes specified axes, or all configured axes if no axis words are supplied.
Example: STEPPER GC G1 X10 Y5 F500

```
STEPPER GS string$
```

Executes a G-code command supplied as a string expression. Accepts any MMBasic string variable, literal, or expression and passes it to the same G-code parser as STEPPER GC.
This is useful when G-code lines are constructed dynamically at runtime.
Supported codes: G0, G1, G2, G3, G4, G28, G90, G91, G92, M3, M5.
Example: STEPPER GS "G1 X" + STR$(x_pos) + " Y" + STR$(y_pos) + " F500"
Example: STEPPER GS gcode$

```
STEPPER GCODE G0|G1|G2|G3|G4|G28|G90|G91|G92|M3|M5 [, X, x] [, Y, y] [, Z, z] [, F, feed] [, I, i] [, J, j] [, K, k] [, R, r] [, P, ms]
```

Alternative syntax for adding motion commands to the buffer. All parameters must be comma separated.
G4 uses P in milliseconds (for example: STEPPER GCODE G4, P, 500).
Example: STEPPER GCODE G1, X, 10, Y, 5, F, 500

3.2 Manual Positioning

```
STEPPER POSITION HOME
```

Sets all axes to position 0 and clears G92 offsets.

```
STEPPER POSITION axis, position
```

Sets the current position of an axis to the specified value (mm).

4. Advanced Features

```
STEPPER SCURVE 0|1
```

Enables (1) or disables (0) S-curve acceleration profiling for smoother motion.

```
STEPPER JERK value
```

Sets the jerk limit in mm/s^3 for S-curve planning.

```
STEPPER SPINDLE pin [,invert]
```

Configures a spindle control pin used by buffered M3/M5 commands. Pin must not conflict with AXIS/HWLIMITS/E-STOP pins.

5. System Management

```
STEPPER RUN [,0|1]
```

Arms the system and begins executing buffered commands.
Optional mode argument controls behavior when the queue drains:
- 0 (default): remain armed with drivers enabled while idle.
- 1: when idle and no buffered work remains, drivers are disabled and then re-enabled automatically when new queued work arrives.

```
STEPPER CLEAR
```

Clears the G-code buffer (only when motion is idle).

```
STEPPER STATUS
```

Displays detailed system status, including axis positions, buffer state, and configuration (including auto-disable-on-idle mode).

```
STEPPER CLOSE
```

Shuts down the stepper subsystem, releases resources, and deconfigures stepper-owned GPIO pins.

```
PEEK(STEPPER X)
PEEK(STEPPER Y)
PEEK(STEPPER Z)
PEEK(STEPPER ACTIVE)
PEEK(STEPPER STATUS)
PEEK(STEPPER BUFFER)
```

Returns current workspace axis position in mm (X/Y/Z) or active state (ACTIVE returns 1 when stepper is actively processing queued work, 0 when idle, or -1 if the stepper subsystem has not been initialized).
STATUS returns a bitmap of safety and coordinate state:
- bit0: X_MIN limit asserted
- bit1: X_MAX limit asserted
- bit2: Y_MIN limit asserted
- bit3: Y_MAX limit asserted
- bit4: Z_MIN limit asserted
- bit5: Z_MAX limit asserted
- bit6: E-STOP asserted
- bit7: position known (machine coordinates established).

BUFFER returns the number of free slots in the G-code circular buffer. This can be used to throttle G-code submission and avoid overflowing the buffer.