TINY CONTROLS PRIVATE LIMITED
Advanced CNC Systems

TNC-M34

4-Axis CNC Controller

Milling / Routing Motion Control

OPERATION MANUAL

Document No.	TNC-M34-01
Document Rev.	1.0
Product Rev.	1.0
Firmware	V1.3
Date	August 2026

Tiny Controls Private Limited | B-17/A, Nishat Park, Kakrola, New Delhi, India – 110078
www.tinycontrols.com | anil@tinycontrols.com | +91-991-119-3210

Table of Contents

Chapter 1 — Safety

Read this section before connecting the TNC-M34 to a machine, and keep it in mind through installation, commissioning, and everyday operation.

1.1 Emergency Stop and Limit Protection

> ⚠ **WARNING**
> The external E-STOP chain must remove power from the drives by itself. The controller's E-STOP input only stops motion — it does not make the machine safe.

> ⚠ **WARNING**
> The motion kill path runs on the Motion core, not the user-interface core, so it keeps working even if the screen freezes or the UI is busy. A limit or E-stop input stops motion within a few milliseconds regardless of the display, and no program, macro, or probe cycle can override it.

> ⚠ **WARNING**
> Limit override (Limit-O) suspends limit protection on purpose, so a tripped axis can still be jogged clear. Nothing else about the machine is safer while it's active — use it only to clear a tripped limit, then release it immediately.

> ⚠ **WARNING**
> While a limit or E-STOP input reads active, the controller refuses any new motion — jog, MDI, macro, or program — and shows an amber LIMIT HIT chip on the STATE panel. This is what stops the next move from driving further into the switch. Limit-O is the way out; E-STOP has no override.

1.2 Motion Safety

> ⚠ **WARNING**
> Homing drives each axis into its home switch until it trips, then backs off and re-approaches slowly to set the reference position. Set the homing feedrate — a few hundred to a couple of thousand mm/min, depending on the mechanics — to a value the machine can safely absorb on first contact, and confirm this before the first homing cycle on a new or rebuilt machine.

> ⚠ **WARNING**
> Soft travel limits only take effect once an axis has been referenced. Before that, the controller has no idea where the axis physically is, and an unreferenced machine will jog or run straight past its travel bounds. Always home the machine, or reference it in place, before relying on soft limits.

Chapter 2 — Product Overview

The TNC-M34 is a 4-axis CNC controller for milling, routing, and similar motion-control applications, built around firmware V1.3. It runs as an independent motion and I/O controller: connect the drives, switches, and spindle, and it handles jogging, homing, G-code execution, and machine I/O from its own panel.
This manual documents the controller exactly as the shipped firmware behaves, including each panel label's matching name in the configuration files. Where the two differ, both are given side by side. Macro programming — the scripting layer for tool changes, probing routines, and custom sequences — is covered in Part II.

2.1 Key Features

- An 800×480 pixel graphic panel.
- 4 axes of coordinated motion: X, Y, Z, plus a configurable 4th axis (A, B, or C), set up as linear or continuously rotating.
- Full machine and work coordinate system support, including six independently settable work offsets (G54–G59), a G52 local offset for temporary shifts, and a G92 offset for on-the-fly zero setting.
- Switch-based homing on a per-axis basis (configurable feedrate, seek rate, and pull-off distance), with an in-place reference option for machines with no home switches.
- Precision jogging with four selectable step scales (0.001 / 0.01 / 0.1 / 1.0 mm per step), plus a continuous jog mode with its own adjustable velocity.
- A two-pane SD-card / USB file browser with a live 3D toolpath preview and a plain-text program view.
- A capable G-code interpreter, including canned drilling and boring cycles, tool length offsets, and multiple work coordinate systems, compatible with programs from common CAM software.
- Fully user-mapped digital I/O — 16 inputs and 16 outputs — any terminal assignable to any supported function (homing, limits, spindle, coolant, and more) from the MAPPING menu.
- On-board diagnostics showing the live state of all 32 I/O pins at once, plus a per-axis tuning screen for checking motion quality against real hardware.
- A macro scripting language for tool changes, probing routines, safety interlocks, and automatic tool changers — no separate PC or CAM package needed.
- A guided recovery wizard for safely resuming a program after an unexpected power loss.
- Field-upgradeable, encrypted firmware, updating both processor cores together from a standard USB stick.

2.2 Technical Specifications

Parameter	Specification
Display	800×480 pixel graphic panel, 3 fixed zones (status bar, main body, softkey row)
Axes	4 total: X, Y, Z, plus a configurable 4th axis (A/B/C), linear or rotary
Coordinate Systems	Machine coordinates plus Work coordinates (G54–G59), with G52 local offset and G92 offset
Jog Step Scales	0.001 / 0.01 / 0.1 / 1.0 mm per step, plus a continuous jog mode with adjustable velocity

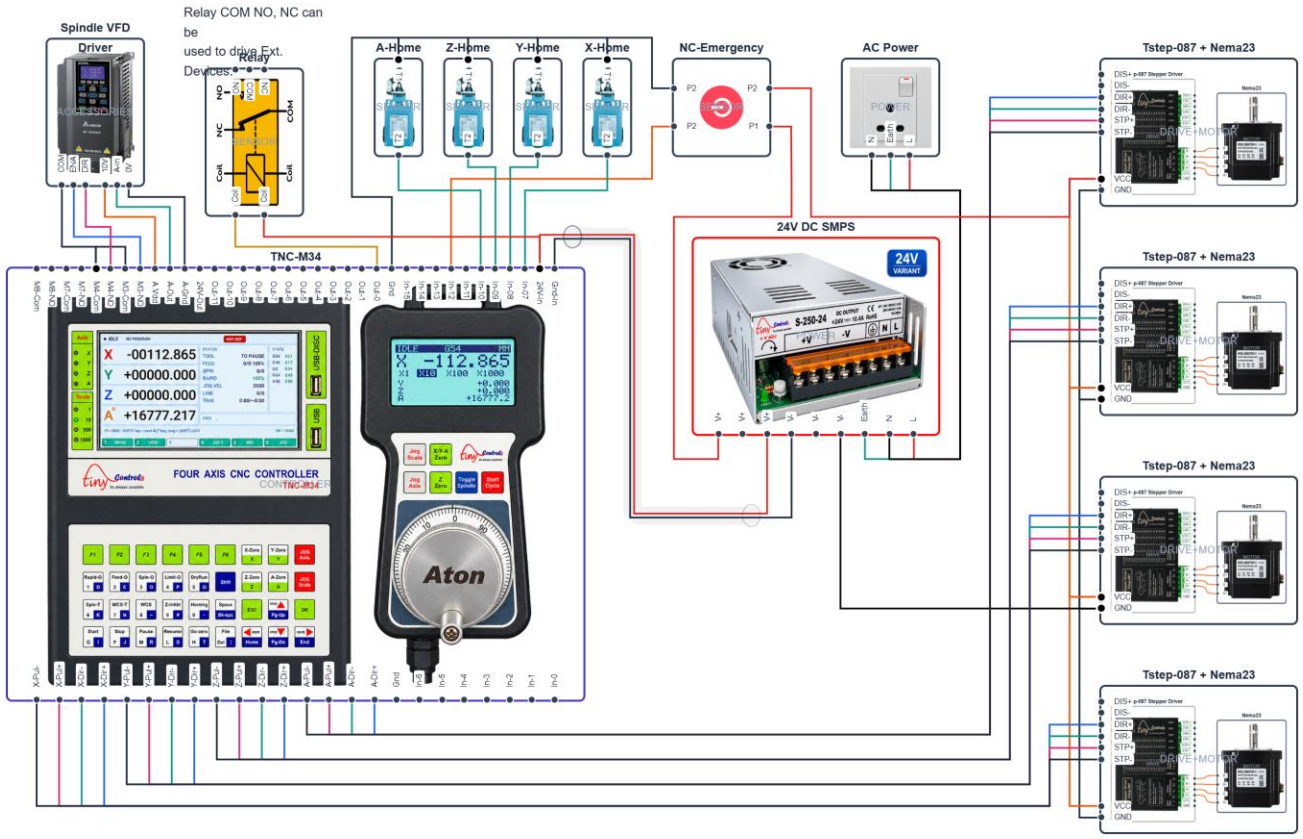
File Storage	Internal SD card (0:/ drive) with USB copy/transfer for loading and backing up programs
I/O	16 digital inputs and 16 digital outputs, each freely assignable to a function in MAPPING
G-code Support	Canned cycles (e.g., G73 peck drilling, G81–G89 drill/bore cycles); tool length and work offsets
Macro Language	Variable- and expression-based scripting for tool changes, probing routines, and interlocks
Firmware	V1.3, running on a dual-core architecture (a user-interface core for the UI, and the Motion core for motion and the safety kill path)
Firmware Update	Encrypted firmware image covering both processor cores, installed via a USB-HOST stick

Chapter 3 — Installation and Wiring

This chapter covers how the controller is wired into a machine: where power goes, how the axis drives, switches, spindle and relays connect, and which terminals carry what. Assigning a function to a particular pin is a separate step, done on the panel — see Chapter 13.

3.1 Electrical Connection Diagram

Figure 3.1 — Electrical Connection Diagram



NOTE

The diagram shows the controller wired into one complete machine — pendant, spindle VFD, an external relay, four home switches, an emergency-stop chain, a 24V supply, and four stepper drive and motor pairs — as a worked example rather than the only valid arrangement. Connect whatever drives, switches and loads the installation actually needs.

3.2 Power Supply

The controller is powered from a 24V DC supply, wired to the 24V-In and Gnd-In terminals. A 24V-Out terminal is provided alongside them for powering field devices such as switches and sensors from the same supply.

3.3 Axis Drive Outputs

Each axis has a differential pulse and direction output pair — X-Pul+/X-Pul− and X-Dir+/X-Dir−, and the same for Y, Z and A. Wire each pair to the corresponding step and direction inputs on that axis's drive. Differential signalling is used because it rejects electrical noise far better than a single-ended output, which matters on a machine where drive cables run alongside motor and spindle wiring.

3.4 Digital Inputs

Sixteen digital inputs are provided, labelled In-0 through In-15 on the terminal strip. Home switches, limit switches, the probe, the emergency-stop chain and physical cycle-start, feed-hold and stop buttons all connect here. No input carries a fixed meaning from the wiring alone — each is given its function, and its polarity, in the I/O mapping menu (Chapter 13).

3.5 Digital Outputs and Relays

Sixteen digital outputs are provided. Twelve are brought out directly on the terminal strip as Out-0 through Out-11; the remaining four drive the on-board relays, each available as a Com and NO contact pair (M3, M4, M7 and M8) so an external device can be switched without additional hardware. As with the inputs, an output does nothing until a function is mapped to it (Chapter 13).

3.6 Spindle Interface

The analog spindle output is brought out as A-Vdd, A-Out and A-Gnd, and wires to the corresponding reference, analog input and common terminals on a VFD to give it a speed reference. Spindle enable and direction are handled separately, as mapped digital outputs driving the VFD's enable and direction inputs.

3.7 Pendant Connection

The handheld pendant connects to the controller by a single cable and provides a jog wheel, axis and scale selection, the zero keys, and cycle start alongside the main panel. It needs no configuration of its own.

Chapter 4 — The Operator Panel

The TNC-M34 is operated entirely from its own panel — no separate PC needed for day-to-day machining. This chapter covers the screen layout, the physical keypad, and the keys whose function you choose yourself.

4.1 Screen Layout

The 800×480 graphic display is divided into three fixed zones, each dedicated to a different kind of information.

Zone	Contents
Top bar	Run-state indicator dot and name, the loaded file, the active work coordinate system (WCS), and any alarm text
Body	The active screen — the main digital readout (DRO), toolpath view, file browser, menus, or configuration pages
Softkey row	Six labelled softkeys (F1…F6) whose function changes to match whatever screen is currently displayed

On the main screen, the softkeys read F1 MENU, F2 VIEW, F3 (assignable — section 4.3), F4 JOG V, F5 MDI, and F6 JOG.
The run-state indicator is the fastest thing to read on the panel:

State	Meaning
IDLE	Nothing is running; all operator actions are allowed
RUN	A program or an MDI block is currently executing
HOLD	Feed hold is active — motion has paused, and the program position is retained
REV	The machine is reversing back along the toolpath from a hold
ALARM	A fault is latched; it must be cleared before anything else can proceed
JOG	A manual jog is in progress

> **NOTE**
> A greyed-out softkey does nothing on the current screen — a deliberate choice, so an unassigned function shows as a visibly dead key rather than a live-looking label that fails when pressed.

4.2 Keypad

The keypad gives direct, one-touch access to jogging, homing, program control, and system functions, without drilling down through menus. Most keys support both a short press and a distinct long press, doubling the functions available from the same buttons.

Key	Short Press	Long Press
F1…F6	The action shown on the softkey above it	Screen-specific (varies by menu)

X-Zero, Y-Zero, Z-Zero, A-Zero	Zero the work offset for that axis at the current position	Open numeric entry to type an exact value for that axis
JOG Axis	Cycle the selected jog axis, X → Y → Z → A	—
JOG Scale	Cycle the jog step size (e.g., 0.001 / 0.01 / 0.1 / 1.0 mm per step)	—
cont LEFT / cont RIGHT	Continuous jog of the selected axis, negative / positive direction	—
step UP / step DOWN	Step jog by the current scale (e.g., 0.1 mm per press); also used for list navigation	—
Homing	Run the homing cycle for all enabled axes	Arm in-place REFERENCE ALL; press Homing again to confirm
Go-zero	Rapid move to the current work zero	—
WCS	Open the work-coordinate system picker (G54–G59)	—
WCS-T	Toggle the DRO between work coordinates and machine coordinates	—
Spin-T	Toggle the spindle on or off	—
Rapid-O, Feed-O, Spin-O	Open numeric entry for that override percentage (e.g., 100%)	—
Limit-O	Suspend limit protection to recover a tripped axis (see Chapter 1)	—
Z-Inhbt	Inhibit Z-axis motion, useful when dry-testing a program	—
DryRun	User-assignable (see section 4.3)	—
File	Open the program browser	—
Start	Cycle start, or resume a held program	—
Pause	Feed hold — pause the running program	—
Stop	Abort the running program	—
Resume	User-assignable (see section 4.3) — resuming a held program is done on Start, not here	A second, independent user-assignable slot
OK	Confirm, open, or apply the current selection	Context-specific (e.g. inside the WCS picker)

ESC	Go back, cancel, or abort the current action	—
Shift	Access the secondary function printed above a key	Shift lock (sticky) until pressed again
Space	Insert a space during text entry	—

The rotary encoder wheel duplicates step UP / step DOWN everywhere, including menus and lists.

> **NOTE**
> During numeric or text entry — MDI, a config edit, an axis value, an override, a file name — the machine-action keys become digits and don't fire their usual action on release. Only Shift-lock stays live while typing.

Each affected key is silk-screened with its digit or Shift-lock character:

Key	Digit (numeric entry)	Shift Character (text entry)
Rapid-O	1	D
Feed-O	2	E
Spin-O	3	O
Limit-O	4	P
DryRun	5	Q
Spin-T	6	K
WCS-T	7	N
WCS	8	–
Z-Inhbt	9	#
Homing	0	.
X-Zero / Y-Zero / Z-Zero / A-Zero	—	X / Y / Z / A respectively
Start / Stop / Pause / Resume	—	I / J / R / S respectively
Go-zero	—	T
File	—	Del
Space	—	Bk-spc (backspace)
cont LEFT / cont RIGHT	—	Home / End respectively
step UP / step DOWN	—	Pg-Up / Pg-Dn respectively

4.3 Assignable Keys

Six keys have no fixed factory job on the main screen, and are configured from CONTROLLER ▸ KEYS (section 12.1a): F3, Space (short and long press), Resume (short and long press), the long press

of Go-zero, and DryRun. Each can run an internal function or the filename of any macro on the card — a tool-zero cycle or a park sequence can be one keypress away without opening a menu.

Panel Key	Notes
F3 on the main screen	Its own short list: NONE, OPT STOP, PARK, AIRBLAST, SAFECHECK, REF POINT, or RUN MAC (default: NONE)
Space (short press)	Short-press-time action for the Space key
Space (long press)	Held-past-threshold action for the Space key
Resume (short press)	Resuming a held program is always done on Start; Resume itself is entirely user-assignable
Resume (long press)	A second, independent user-assignable slot on the same key
Go-zero (long press)	The short press always stays the built-in GO ZERO
DryRun	Assignable like the others below; see the NOTE following this table

Space, Resume (both presses), Go-zero long-press, and DryRun share the same internal function list: NONE (an unassigned long-press falls back to that key's short-press action, so a slightly slow press never lands on a dead key), OPT STOP (toggle M1), GO REF / SET REF (the global reference point, section 7.6), RUN MAC (open the macro picker, no menu PIN needed), BLOCK RUN (toggle single-block mode, section 10.2) — or the filename of any macro on the card, listed live in the chooser.

NOTE

A macro assignment is stored by filename, not by list position, so adding or deleting macros elsewhere on the card can never silently re-point a key. A key naming a deleted macro simply behaves as NONE until a file with that name returns.

NOTE

F3 is automatically taken over by REVERSE whenever the machine is in HOLD, regardless of its configuration, so an operator can always back out of a paused cut the same way. Resume has no built-in action any more — Start has always been the resume key, on both panel and pendant — so both of Resume's presses are free for anything on the list above.

NOTE

DryRun is fully assignable like the others here, and can be set today to any internal function or macro. A dedicated TEACH function is planned for a future firmware update and will become one more option once it arrives.

Chapter 5 — First Power-On

This chapter walks through the recommended sequence for commissioning a new TNC-M34 installation, from first power-up to a fully referenced, ready-to-run machine. The SD card is factory-installed and pre-loaded — no card handling required.

5.1 Initial Setup Sequence

1. On the very first boot, the firmware writes a complete set of default configuration files to 0:/config/ and continues booting normally. Every default value can then be reviewed and edited from the panel.
2. The backlight stays dim until the first frame is drawn, so a slower-than-usual boot shows a dim panel rather than a bright, blank one.
3. Go to MENU ▸ MACHINE and set the motion parameters in this order: axis steps/mm (e.g., 1000 steps/mm for a typical leadscrew), travel limits, maximum feed rates and accelerations, then homing. Getting these right first avoids chasing confusing symptoms later.
4. Go to MENU ▸ MAPPING and assign every physical input and output actually wired to the machine. Nothing is assigned by default beyond a minimal starting map — this step turns raw wiring into working homing switches, limit switches, and spindle control.
5. Home the machine using the switches, or reference it in place if it has none (section 7.4), so the controller has an accurate, repeatable idea of where the axes physically are before any cutting begins.

5.2 Files on the SD Card

All configuration, programs, and macros for the TNC-M34 live on its internal SD card, in a small, predictable set of folders and files:

Path	Contents
0:/config/machine.cfg	Axis parameters, homing, spindle, travel limits, tool table, and 4th-axis settings
0:/config/controller.cfg	UI behaviour, key assignments, and display brightness settings
0:/config/stock.cfg	Stock shape and size, used only to draw the 3D toolpath preview
0:/config/io.cfg	The complete input/output function mapping matrix
0:/config/offsets.cfg	The G54–G59 work offsets and the tool length offset table
0:/gcode/	G-code programs — only files stored in this folder can be run
0:/macros/	.mac macro files, used for tool changes, probing, and custom routines

> **NOTE**
> All config files can be opened and edited directly on the panel — no PC needed for routine setup. A program stored outside 0:/gcode can still be viewed, but not run: the FILES screen loads it read-only rather than listing something it can’t execute.

Chapter 6 — Coordinate Systems

Understanding the relationship between machine coordinates and work coordinates is one of the most important things for getting reliable, repeatable results from the TNC-M34. This chapter explains both coordinate systems, how the offsets combine, and how to set them from the panel.

6.1 Machine and Work Coordinates

Machine coordinates are fixed to the physical machine and only mean anything once an axis has been referenced — before that, they're simply undefined. Work coordinates are the coordinate system a G-code program is actually written in, and what most operators think of as "the" position day to day. WCS-T toggles the DRO between the two views, and the display always makes clear which one is showing. The DRO's work reading is simply the machine position minus the active work coordinate system offset, and nothing else:

```
DRO work coordinate = machine coordinate - G54..G59 offset
```

The other frame shifts — G92, the G52 local offset, and the G43 tool length offset — are applied by the parser to whatever coordinate a program or MDI line actually commands, not to the position the DRO displays:

```
commanded position = word + G54..G59 offset
                          + G92 offset
                          + G52 local offset
                          + tool length offset (G43, Z only)
```

The two frames agree whenever G92, G52, and G43 are all clear — the normal case. When one is active they no longer agree: the DRO keeps reading against the work coordinate system alone, while the program actually cuts at the shifted position. This matters most for touch-off — see the tool-offset note in section 6.4 and the #tlo variable in the macro reference (Chapter 28). workzero.mac deliberately refuses to file a reference position while a tool offset is in force, rather than produce a number the DRO agrees with and the program doesn't.

NOTE

Coordinate range: ±16777.216. Positions are stored as single-precision floats, so only values below this bound hold three decimal places exactly — 16777216 is 2^{24}, the largest integer a float can count to, and in thousandths that's 16777.216 units. A typed coordinate beyond the limit is clamped, with the limit shown on screen (e.g., X LIMIT 16777.216), rather than silently rounding to the nearest representable value.

6.2 Work Offsets G54–G59

Press WCS to open the work coordinate system picker. The active system is shown filled in, and the current selection is highlighted with a ring.

- OK selects the highlighted system as the new active work coordinate system.
- A long press of OK both selects the system and rapids the machine to its stored zero, saving a separate Go-Zero step.
- WCS, or the Up/Down keys, cycles the selection through G54 to G59.

NOTE

All six work offsets are stored in 0:/config/offsets.cfg and survive a power-off, so a machine doesn't need re-zeroing every time it's switched on.

> ⚠ **WARNING**
> Changing the active work coordinate system re-parses a loaded program if that program names its own G54–G59. A file's coordinates are resolved into whichever frame was active when it was loaded, so a file written in G56 while the machine sits on G54 has every coordinate pre-shifted by the difference (the extents display flags this in amber). That shift is only a snapshot taken at load time, so switching frames afterward means the controller simply re-reads the file — taking as long as the original load did. A file that never names a work coordinate system is unaffected either way.

6.3 Setting Work Zero

With the tool positioned exactly where zero should be, press the corresponding axis Zero key (X-Zero, Y-Zero, Z-Zero, or A-Zero). That axis's work offset is calculated and stored automatically so the DRO immediately reads zero at the current position. To enter an exact numeric coordinate instead, long-press the same key to open numeric entry — the short-press zero action is skipped, so a long press never does both.

> **NOTE**
> The workzero.mac macro, provided by default in the macro folder, sets the currently selected work coordinate system to zero at the probed position, using the G38 probing cycle and storing the result with WORKZERO (see section 18.6).

6.4 Tool Offsets

G43 Hn applies tool n's stored length offset to the Z axis, compensating for tools of different lengths; G49 cancels the active offset. The tool offset table lives in offsets.cfg and can be written directly from the panel, from MDI with G10 L1 Ph Zv, or by a probing macro's TOOLLEN verb.

> **NOTE**
> Automatic tool-length touch-off is supported through the toolzero.mac macro, provided by default in the macro folder. It drives the G38 probing cycle and stores the result directly into the tool offset table with TOOLLEN (section 18.6).

> ⚠ **WARNING**
> G43 shifts what the parser commands, not what the DRO shows (section 6.1). Always probe a work reference position with G49 in force — workzero.mac refuses otherwise, rather than file a number the DRO agrees with while the program cuts at a different, offset position.

6.5 Reset

MENU ▸ SYSTEM ▸ RESET opens a chooser between two very different resets. Each shows exactly what it's about to do before it's armed, and each needs an extra OK to confirm.

- RESET COORDINATES declares the machine's current position to be machine zero and clears all six work offsets (G54–G59) back to zero. The tool offset table, other settings, the menu PIN, and the licence are all left untouched. This exists for machines with no home switches, where there's otherwise no way to establish a defined origin.
- RESET ALL TO FACTORY returns every setting to its shipped default — axes, homing, the I/O map, key assignments, the tool offset table, and the stock display settings — clears the coordinates and the stored global reference point, and resets the menu PIN to 123456. Two things survive a factory reset: the licence (a factory reset un-personalises the setup, not what

was paid for) and the clock. Re-home the machine afterward, since its stored geometry has gone back to defaults.

> ⚠ **WARNING**
> Because both resets discard real setup data — RESET ALL TO FACTORY especially — confirmation on each is deliberately strict: review what the screen says, then press OK a second time to apply it.

Chapter 7 — Referencing and Homing

Homing establishes a known, repeatable reference position for each axis, which every other coordinate on the machine is measured from. This chapter explains the two coordinates that define homing for each axis, walks through the homing cycle, lists every related setting, and covers the special cases — safe-Z lifts, shared switches, and machines with no switches at all.

7.1 The Homing Cycle

Each axis homes through the same five-step sequence, in the order set by its homing sequence number (axes sharing a sequence number run together as a group), followed by a single parking move for the whole cycle:

1. Pre-check — the switch state is read before any motion begins. If already tripped, the axis releases it first rather than seeking.
2. SEEK — the axis moves toward the switch at the configured homing feedrate (e.g., 2000 mm/min) until it trips.
3. Release — the axis reverses until the switch opens again, then backs off by the configured sensor-clear margin (e.g., 2 mm).
4. RESEEK — the axis approaches the switch a second time, at the slower homing seek rate (e.g., 500 mm/min), for a repeatable trip point.
5. Release again, then stop — the reference position is stamped the instant the axis comes to a complete standstill, clear of the switch. This is the switch coordinate from then on.
6. Park — happens once, after every axis in the cycle has been referenced, not per axis at its own reference position. X, Y, and the 4th axis (whichever are in the cycle) travel to their park coordinates together, in one rapid move; Z parks last, alone, once that move finishes — Z's park is the one move that can bring the tool back into the work envelope, so nothing else may still be moving when it happens. An unreachable or mis-typed park is refused, and named, at that axis's own reference position, where its seek direction is known.

NOTE
Soft travel limits are suspended for the whole homing cycle, since home switches legitimately sit at or beyond the normal soft travel range. The cycle's own overtravel guards bound the motion instead, and soft limits re-engage the moment the cycle ends or aborts.

NOTE
The reference position is always taken after the slow reseek pass, with the axis fully at rest, approaching at the same speed every cycle — this is what makes the standstill position as repeatable as the switch itself.

7.2 Homing Settings

Panel	Config Key	Notes
Home X in cycle (switch)	homing/enable_x	NO / YES — whether this axis participates in the homing cycle at all
X seek direction	homing/invert_x	TOWARD − / TOWARD +. This sets direction only, not switch polarity — polarity (NO/NC) is set separately in MAPPING

X switch coord	homing/pos_x	The machine coordinate the axis is assigned when the homing cycle establishes its reference position, just clear of the switch (e.g., 0.000 mm)
X park coord (after home)	homing/park_x	The machine coordinate the axis moves to once every axis in the cycle has been referenced. No axis parks until then: X/Y and the 4th travel to their parks together in one rapid, and Z parks last, alone (section 7.1). A mis-set park is refused, and named, at that axis's own reference position
X homing seq (1–4)	homingseq/seq_x	Ascending order; axes sharing a number home together as a group. Give Z the lowest number of the group if the safe-Z lift (below) is enabled — see the warning in section 7.3
Search cycles	homing/search_cycles	Number of seek/release repetitions used to confirm the switch (e.g., 2)
Debounce ms	homing/debounce_ms	Switch debounce filter time (e.g., 10 ms)
Sensor clear	homing/sensor_clear	Back-off margin applied after each release, common to all axes (e.g., 2 mm)
Feedrate	homing/feedrate	Speed of the first, coarse seek toward the switch (e.g., 2000 mm/min)
Seekrate	homing/seekrate	Speed of the fine reseek and all release moves (e.g., 500 mm/min)
Z safe	homing/z_safe	Work-frame Z height to lift to before homing X/Y (and for GO ZERO / GO REF); skipped entirely when Z is already at or above it — capped for safety, see section 7.3 (e.g., 50 mm)
Lift Z before homing X/Y	homing/z_safe_enable	Whether the Z-safe lift is performed at all. It is spliced in immediately after Z's own homing entry, so it only protects axes sequenced after Z — see section 7.3
GZ lift enable	homing/gz_lift_enable	When enabled, GO ZERO lifts to the Z-safe height first, before moving X/Y
Abort retract	homing/abort_retract	When enabled, pressing STOP rapids Z to the safe height before halting

7.3 Safe-Z and Overtravel

Z safe is read as a work-frame height, not a machine coordinate: a value of +50 means 50 above the current work zero — a statement about the setup, which is where a safe height belongs. (Earlier firmware read it as a machine coordinate; with the shipped default of +50 against a Z travel_max of 0,

every lift clamped straight to machine zero, so Z retracted all the way to the top of travel on every homing or GO ZERO leg regardless of the setup.) The lift is skipped entirely whenever Z already sits at or above the computed target.
Once converted to a machine height, the target is clamped to a hard ceiling:

- The Z axis travel_max — the configured top of travel, whatever its sign.
- When Z seeks TOWARD + (switch mounted above), one sensor-clear margin below the Z switch coordinate too.

CAUTION
An axis is never commanded back through the switch that referenced it, whatever the configuration says. When the ceiling bites, the panel reports SAFE Z (work <n>) capped - using <n> — this cap applies consistently to the homing lift, GO ZERO, GO REF, the abort retract, and the Start approach move (section 10.1).

⚠ **WARNING**
travel_max must genuinely be the real top of Z travel. If left at or below travel_min, the axis reads as unconfigured and the ceiling falls back to machine zero — on a machine with positive Z coordinates, that pins every safe-Z target at 0, below where Z already sits. Since a lift is skipped whenever Z is already above the target, no lift ever happens: no abort retract, no GO ZERO lift, no Start approach. Double-check travel_max against the actual machine before relying on any safe-Z behaviour.

A skipped lift now says so on screen — ABORT: no retract - Z <n> is above safe <n>, or GO ZERO (no lift - Z already above safe Z) — so "the retract is off" and "the retract had nothing to do" can be told apart at a glance.

NOTE
The safe-Z lift depends on where Z sits in the homing sequence. It’s spliced into the queue right after Z’s own homing entry, so it isn’t inserted at all when another axis is sequenced before Z (that axis seeks at whatever height the tool happens to be at), when Z shares its sequence number with another axis (they home together — no Z-then-XY boundary to lift at), when nothing is sequenced after Z, or when Z homing is disabled entirely (referenced in place, section 7.4 — no Z entry for the lift to attach to, so X/Y seek at whatever height the tool is at). In every one of these cases, Z safe enable still reads ON and simply does nothing; the HOMING SEQ page flags this in amber, and the fix is almost always to give Z the lowest sequence number in the group.

Between the switch tripping and the axis actually stopping, frame latency, debounce filtering, queued motion, and deceleration distance add up to a small amount of overtravel past the trip point. At the default first-seek rate of 2000 mm/min, expect roughly 6 mm; the fine reseek at 500 mm/min — the pass that sets the reference position — is much smaller, around 1 mm. If the initial contact is hard enough to risk skipping steps, lower the homing feedrate — deceleration distance scales with the square of velocity, so even a modest reduction buys a much gentler first contact.

7.4 Referencing Without Switches

Long-press Homing to arm in-place REFERENCE ALL, then press Homing again to confirm. Every axis is immediately declared referenced exactly where it stands, with no motion involved. Use this on a machine with no home switches, or to re-establish reference after a recovery once the machine has been repositioned by hand.
Any axis with homing disabled in configuration is automatically referenced in place at the start of a homing cycle, so a mixed machine — switches on X and Y but none on Z, say — homes correctly in a single pass with no extra manual steps.

7.5 Shared Switches

If two axes are mapped to the same physical input (to save wiring on a small machine, say), they're automatically homed one at a time rather than as a group, regardless of their sequence numbers. This stops one axis from mistaking the other axis's switch trip for its own.

7.6 Global Reference Point

When F3 is assigned to REF POINT (section 4.3), these functions become available on that key:

- SET REFERENCE HERE — stores the current machine position as the global reference point.
- GO TO REFERENCE — rapids the machine back to the stored reference position.
- CLEAR REFERENCE — deletes the stored reference point.

The reference point is stored in battery-backed RAM and survives a full power-off. It's a convenient way to mark a known, safe position before an operation — a manual axis move, say — that might otherwise cause the machine to lose its reference.

Chapter 8 — Jogging

Jogging is manual movement of an axis for setup, tool changes, and general positioning. The TNC-M34 offers both precise step jogging and fast continuous jogging, from the same set of keys:

1. Press JOG (F6) or JOG Axis to arm jogging. The softkey label changes to JOG* while armed, and the controller remembers the last axis selected.
2. JOG Axis cycles the selected axis through X → Y → Z → A, one press at a time.
3. JOG Scale cycles the step size through 0.001 / 0.01 / 0.1 / 1.0 mm per step.
4. step UP / step DOWN, or the rotary encoder, moves the selected axis by exactly one step at the current scale.
5. cont LEFT / cont RIGHT jog the selected axis continuously while held.
6. F4 (JOG V) opens numeric entry to set the continuous jog velocity in mm/min (e.g., 2000 mm/min). This value is stored as ui/jog_vel in CONTROLLER config, and shown in the status line.

> **NOTE**
> Continuous jog velocity is capped by the axis's own configured maximum rate — requesting more than the axis can do just gives the axis maximum, never an error.

8.1 Wheel Feel

Detent jogging — from the pendant MPG or the panel's own encoder wheel — follows the wheel proportionally rather than stepping one fixed distance per click. Commanded speed is the accumulated turning backlog divided by a time constant, capped by the axis's own limits; that constant is CONTROLLER ▸ INTERFACE ▸ Wheel follow time s (ui/mpg_follow, 0.02–0.5 s, default 0.10 s).
A steady crank rolls the axis at exactly the crank rate: a feed-forward term commands the measured crank rate directly, so the backlog only trims the result, and steady-state lag works out to roughly a tenth of the crank rate times the time constant. The setting is the feel knob — smaller feels tighter and twitchier, larger feels smoother but laggier. Cranking faster speeds the axis up smoothly; stopping the wheel decays speed with the backlog down to an exact landing, no per-detent sprint-and-brake, no surging. Reversing the wheel stops the axis at full acceleration first, so direction changes stay crisp, and a single detent still lands briskly thanks to a small floor speed that cuts off the tail. Held-key continuous jogging is unaffected.

Chapter 9 — Files

Press File, or use MENU and navigate to it, to reach the FILES screen — where G-code programs are browsed, transferred, and loaded for running. It's a two-pane browser, SD card on the left and a connected USB drive on the right; the continuous-jog left/right keys move the highlighted focus between panes.

Key	Action
cont LEFT / cont RIGHT	Move the focus between the SD and USB panes
F2	BACK — return to the previous screen
F3	Copy the selected file from SD → USB
F4	Copy the selected file from USB → SD
F5	DELETE the selected file (press again to confirm)
F6	CLOSE the file browser and return to the main screen
OK	Load the selected program
Up/Down, wheel	Navigate the file list

NOTE
Long file names are supported up to 96 characters. A file that can't be opened is simply never listed — if it appears in the browser, it can be opened.

NOTE
Copying a large file shows a progress bar and keeps the panel fully responsive; a 30 MB program takes a noticeable while to copy, but that's normal, not a hang.

9.1 Loading a Program

Loading a program jumps straight to the VIEW screen so the operator can inspect it immediately. For a file that isn't runnable (stored outside 0:/gcode), the view opens on the TEXT tab instead of the toolpath, since there's no program to preview graphically.
Very large programs load in progressively reduced tiers as size demands — first dropping the text editor, then detailed modal-state tracking, finally falling back to a scan-only tier that still gives a toolpath preview and line count but no editing. The active tier is shown on screen during the load, so it's never a surprise. A 30 MB program typically loads in roughly 12 seconds.

9.2 Viewing and Editing

The VIEW screen has two tabs, switched with F1:

- Toolpath — a full 3D preview of the program. F5 zooms in; a long press of F5 fits the entire path on screen. The camera (azimuth, elevation, zoom, and pan in X and Y) is selected with F1/F2/etc. and nudged with the rotary wheel. The tool position is drawn as a large, contrasting marker, easy to find even in a dense toolpath.
- Text — the raw program source, with a built-in editor whenever the load tier allows it. F6 arms GOTO, for jumping to a specific line number.

Chapter 10 — Running a Program

This chapter covers starting, pausing, and stopping a running program, adjusting feed and spindle overrides on the fly, resuming from a specific line, and configuring optional stops and tool changes.

10.1 Start, Hold, Stop

Key	Action
Start	Begin the loaded program from the top, or resume a program that is on hold
Pause	Feed hold — decelerate smoothly and stop, retaining the exact program position
Resume	User-assignable (section 4.3) — resuming a held program is always done on Start
Stop	Abort the running program. If abort_retract is enabled, Z rapids to the safe height first
F3 (while HOLD)	REVERSE — retrace back along the toolpath already travelled

NOTE
These same four actions also exist as physical inputs (CYCLE START, FEED HOLD, STOP) when mapped in MAPPING, for machines with dedicated physical buttons. The physical buttons behave identically to their panel-key equivalents.

Pressing Start on an idle machine doesn't jump straight to the program's first commanded move — it drives the approach in a fixed, safe order. Z lifts first, to a height derived from the file itself (its highest programmed Z, never below wherever Z already is, clamped at machine zero); X, Y, and the 4th axis then traverse; Z comes down last, at whatever rapid or feed rate the first move specifies, never faster. The plan also begins at the file's own first commanded coordinates rather than work zero, so an axis the program never mentions doesn't move at all. Two kinds of file keep the older, more conservative behaviour on Z instead: one whose first Z action is a downward cut with no clearance move above it, and one whose first Z move appears inside a helix. For either, adding an explicit clearance height as the file's first Z move — which most post-processors already do — restores the normal approach.

NOTE
This approach sequence belongs to Start alone. MDI lines, macro moves, and the tuning calibration move are always literal — G0 Z-5 goes straight to Z-5 from wherever the machine stands, none of the lift-then-traverse-then-descend logic above. RUN FROM LINE (section 10.4) uses this same approach sequence, just starting from a different line.

10.2 Block Run (Single-Block Mode)

BLOCK RUN makes a running program stop at the end of every source line instead of running straight through: one press of Start runs exactly one line, then the machine holds; the next Start runs the next line. It's the mode for proving out a new program, or a new fixture, at the machine before trusting it to run unattended.

- Turn it on from MACHINE CONFIG ▸ GENERAL ▸ block run (section 12.3), from an assignable key set to BLOCK RUN (section 4.3), or from a mapped BLOCK RUN input (section 13.1) — all three toggle the same mode.
- The step key is Start — the same key that resumes from any other hold. There's no separate step key or step input.
- An amber BLK RUN chip stays on the STATE panel while the mode is active, so a program stopping on every line never reads as a machine that keeps stalling.
- Every line boundary becomes an exact stop while the mode is active — the motion planner zeroes the corner speed there — so the tool genuinely stops on the line's end point, not just past it. A contoured path run this way stops at every one of its points, so expect witness marks if the cut actually runs in this mode.
- One line means one line however many moves it expands into: an arc's tessellated segments, and the Start approach legs above, each belong to their own line and run as one step.
- A line with no motion — an M-code alone, or a comment — isn't a step of its own; it takes effect with the next moving line.

Switching the mode on mid-run is fine: the machine decelerates into the first line boundary it reaches (it may carry a little past it, since the corner was already planned assuming it would keep going), and every step after that is exact. Switching it off while held just means the next Start runs the program to completion as normal. HOLD, M0/M1 optional stops, tool-change pauses, and REVERSE all work normally alongside Block Run.

10.3 Overrides

Feed-O, Rapid-O, and Spin-O each open a numeric entry box; OK applies the new value immediately. All three are percentages of the programmed rate (100% = programmed, 150% = 50% faster).

> ⚠ **WARNING**
> Two settings in CONTROLLER config, ui/fixed_feed and ui/fixed_rapid, are forces rather than ceilings: set either to a non-zero value and it replaces the programmed feed or rapid rate outright, regardless of what the program asks for or what override the operator has dialled in. 0 (the default for both) restores normal, programmed-rate behaviour. Because this actively overrides the program rather than just limiting it, treat a non-zero value as a deliberate, setup-specific choice, not a general-purpose safety ceiling.

10.4 Run From Line

On the main screen, long-press Start, type the line number, and press OK to start the program from there.

10.5 Optional Stop and Tool Change

- M1 optional stop — when enabled, an M1 command pauses execution and waits for the operator; when disabled, M1 is ignored.
- M6 tool change — determines what happens on M6: IGNORE (skip it), PAUSE (stop and wait for Start once the tool's been changed by hand), or MACRO (run the configured tool-change macro).

Chapter 11 — MDI

Manual Data Input (MDI) lets the operator execute a single G-code block immediately, without loading a full program — useful for quick manual moves, testing a command, or one-off setup steps. Press F5 on the main screen to open MDI, type a G-code block, and press OK. MDI blocks are modal just like a full program: G91 G1 X5 F200 followed by X5 performs the second move using the same modal settings as the first, without repeating G91, G1, or F200.

NOTE

MDI is refused whenever the machine is not idle and unlocked, preventing a stray command from interfering with a program that is already running.

⚠ **WARNING**

A feed move needs a feed. G1, G2, G3, and canned-cycle points with no F currently in effect are refused outright — UNSUPPORTED G1 WITHOUT F — rather than actually run: internally, F=0 is treated as "no feed limit at all," so such a move would otherwise execute at the axis's full rapid rate. The same rule is enforced when a program is loaded (the file is refused, naming the offending line) and inside a macro.

NOTE

MDI's modal state — the last-typed F, G90/G91, G92, and the active tool offset — belongs to the operator alone; macros no longer share it. Each macro run gets its own freshly re-initialised modal context, so anything a macro sets internally never leaks back into MDI, and MDI's own settings survive any macro run untouched.

Chapter 12 — Menus and Settings

The TNC-M34 has a detailed settings menu covering everything from motion tuning to display brightness, organized into categories, subcategories, and individual parameters. This chapter explains how to navigate it and lists every setting, its config-file key, and what it controls.

12.1 Menu Structure

MENU (F1 from the main screen) opens the top-level menu, containing five main categories:

Entry	Contents
CONTROLLER	UI behaviour, themes, and display brightness, the KEYS page for all six user-assignable keys (section 4.3), plus the CLOCK and SECURITY pages (section 12.7)
MACHINE	Axes, homing, spindle, travel limits, tool table, and 4th-axis settings
MACROS	Browse and run .mac files
MAPPING	The complete input/output function mapping matrix
SYSTEM	ABOUT, HELP, DIAGNOSTICS, TUNING, PENDANT UPDATE, and RESET (coordinates, or all settings to factory)

Navigation is consistent throughout: wheel or Up/Down to scroll, OK to open a category or select a parameter, ESC to step back up one level, F6 to close straight to the main screen.

12.2 Complete Menu Tree

Every menu page, every field on it, and the values each field accepts. The ranges below are the ones the controller actually enforces — the same table the file loader validates against — so a range printed here cannot disagree with what the panel will let you enter.
Rows the panel hides permanently are not listed. Rows it hides conditionally are: the 4-AXIS travel limits disappear while the axis is set to roll over or run continuously, and rollover and continuous themselves disappear when the axis is LINEAR — but all of them are always present in the configuration file.

```
MAIN SCREEN -- F1 (MENU, PIN-gated)
|
+- CONTROLLER ------------ controller.cfg
|   |
|   +- INTERFACE
|   |    theme         DARK / LIGHT
|   |    accent        TEAL / AMBER / VIOLET / GREEN
|   |    opt_stop      OFF / ON
|   |    tool_change   IGNORE / PAUSE / MACRO
|   |    fixed_feed    0 .. 200000  mm/min
|   |    fixed_rapid   0 .. 200000  mm/min
|   |    jog_vel       1 .. 200000  mm/min
|   |    mpg_follow    0.02 .. 0.5  s
|   |
|   +- DISPLAY
|   |    lcd_backlight      0 .. 100  %
|   |    pendant_backlight  0 .. 100  %
|   |    pend_contrast      -1 .. 63
|   |    pend_ratio         0 .. 7
|   |    led_axis_bright    0 .. 100  %
```

```
|   |    led_scale_bright   0 .. 100  %
|   |
|   +- CLOCK
|   |    year    2000 .. 2099
|   |    month   1 .. 12
|   |    day     1 .. 31
|   |    hour    0 .. 23
|   |    minute  0 .. 59
|   |    second  0 .. 59
|   |
|   +- KEYS
|   |    All six assignable rows below take the same list:
|   |      NONE / OPT STOP / GO REF / SET REF / RUN MAC /
|   |      BLOCK RUN / <any .mac file on the card>
|   |
|   |    space_key      f3_fn takes its own shorter list:
|   |    space_hold       NONE / OPT STOP / PARK / AIRBLAST /
|   |    resume_key       SAFECHECK / REF POINT / RUN MAC
|   |    resume_hold
|   |    gozero_hold
|   |    dryrun_key
|   |    f3_fn
|
+- MACHINE --------------- machine.cfg
|   |
|   +- GENERAL
|   |    step_pulse_us  1 .. 100  us
|   |    step_delay_us  0 .. 100  us
|   |    junction_dev   0.0001 .. 10  mm
|   |    arc_tol        0.0001 .. 10  mm
|   |    smooth_time    0 .. 0.5  s
|   |    motion_mode    G64 BLEND / G61 EXACT
|   |    block_run      OFF / ON
|   |
|   +- X-AXIS   (Y-AXIS and Z-AXIS are identical)
|   |    steps_per_mm  1 .. 100000  steps/mm
|   |    max_rate      1 .. 200000  mm/min
|   |    accel         1 .. 200000  mm/s2
|   |    travel_min    -100000 .. 100000  mm
|   |    travel_max    -100000 .. 100000  mm
|   |    invert        NORMAL / REVERSED
|   |    flip jog      NORMAL / FLIPPED
|   |    backlash      0 .. 100  mm
|   |
|   +- 4-AXIS
|   |    type           ROTARY / LINEAR
|   |    rollover       OFF / ON
|   |    letter         A / B / C
|   |    parallel       X / Y / Z
|   |    steps_per_deg  1 .. 100000  steps/deg (per mm if LINEAR)
|   |    max_rate       1 .. 200000  deg/min (mm/min if LINEAR)
|   |    accel          1 .. 200000  deg/s2 (mm/s2 if LINEAR)
|   |    continuous     OFF / ON
|   |    travel_min     -100000 .. 100000  deg (mm if LINEAR)
|   |    travel_max     -100000 .. 100000  deg (mm if LINEAR)
|   |    invert         NORMAL / REVERSED
|   |    flip_jog       NORMAL / FLIPPED
|   |    backlash       0 .. 100  deg (mm if LINEAR)
|   |
```

```
|   +- HOMING
|   |    enable_x        NO / YES        (same four rows for
|   |    invert_x        TOWARD - / +     y, z and a)
|   |    park_x          -100000 .. 100000  mm
|   |    pos_x           -100000 .. 100000  mm
|   |    search_cycles   1 .. 10
|   |    debounce_ms     0 .. 1000  ms
|   |    sensor_clear    0 .. 1000  mm
|   |    feedrate        1 .. 200000  mm/min
|   |    seekrate        1 .. 200000  mm/min
|   |    z_safe          -100000 .. 100000  mm
|   |    z_safe_enable   OFF / ON
|   |    gz_lift_enable  OFF / ON
|   |    abort_retract   OFF / ON
|   |
|   +- HOMING SEQ
|   |    seq_x  1 .. 4      seq_z  1 .. 4
|   |    seq_y  1 .. 4      seq_a  1 .. 4
|   |
|   +- LIMITS
|   |    hard  OFF / ON
|   |    soft  OFF / ON
|   |
|   +- SPINDLE
|   |    on_delay      0 .. 600  s
|   |    mode          0-10V / PWM
|   |    min_rpm       0 .. 300000  rpm
|   |    max_rpm       0 .. 300000  rpm
|   |    pwm_freq      1 .. 1000000  Hz
|   |    pwm_min_duty  0 .. 100  %
|   |    pwm_max_duty  0 .. 100  %
|   |
|   +- TOOL ZERO
|   |    sensor_height  -100000 .. 100000  mm
|   |    backoff        -100000 .. 100000  mm
|   |    distance       -100000 .. 100000  mm
|   |    feed           1 .. 200000  mm/min
|   |    seek           1 .. 200000  mm/min
|   |    pulloff        -100000 .. 100000  mm
|   |    use_pos        OFF / ON
|   |    setter_x       -100000 .. 100000  mm
|   |    setter y       -100000 .. 100000  mm
|
+- MAPPING -------------- io.cfg
|   |
|   +- MAPPING   two rows per I/O function (13.1, 13.2)
|   |    in_<function>      -1 .. 15  (-1 = unassigned; pin number)
|   |    in_<function>_nc   OFF / ON  (ON = normally closed)
|   |    out_<function>     -1 .. 15  (-1 = unassigned; pin number)
|   |    out_<function>_lo  OFF / ON  (ON = active low)
|
+- MACROS --------------- 0:/macros/*.mac -- select, OK runs
|                         (also PIN-free via F3 = RUN MAC)
|
+- SYSTEM
    +- ABOUT             versions, DEVICE ID, pendant found
    +- HELP              scrollable on-panel reference
    +- DIAGNOSTICS       live pins, force outputs, LIM OVR
    +- TUNING            steps/mm + backlash wizards
```

```
    +- PENDANT UPDATE    Pendant V2 over RS485
    +- RESET             coordinates, or ALL to factory
```

STOCK settings are not in the menu at all — they are edited in stock.cfg on the card:

```
STOCK (stock.cfg)
    enable    OFF / ON
    shape     BOX / CYLINDER
    mode      AUTO / MANUAL
    zref      TOP / BOTTOM
    margin    0 .. 10000  mm
    size_x    -100000 .. 100000  mm
    size_y    -100000 .. 100000  mm
    size_z    -100000 .. 100000  mm
    origin_x  -100000 .. 100000  mm
    origin_y  -100000 .. 100000  mm
    origin_z  -100000 .. 100000  mm
    diameter  0 .. 100000  mm
    length    0 .. 100000  mm
    position  -100000 .. 100000  mm
```

NOTE

Three things are deliberately absent from the menu. Work offsets are written by the ZERO keys, by G10 L2, by G92, and by the probing macros. The tool table is written by G10 L1 P<h> Z<v> or by a macro's TOOLLEN verb, and read back by G43 Hn. Stock display settings live in stock.cfg on the card, as above.

12.3 Editing a Setting

Select a row and press OK to begin editing.

- Numeric values open a numeric entry box — type the value, then OK to accept it.
- Boolean and small enumerated values toggle directly on OK, cycling through readable labels such as NO/YES, TOWARD −/TOWARD +, DARK/LIGHT, or IGNORE/PAUSE/MACRO, rather than a raw 0 or 1.

NOTE

Edited values are held in RAM until the page is saved. ESC or F2 walks back up through page → group → menu, saving on the way out at each level. A machine action taken without passing through that save chain uses the live in-memory values, but the SD card stays unchanged — so the next reboot silently reloads the previous saved settings.

NOTE

Settings that don't apply to the current configuration are hidden from the menu entirely, not just greyed out — set the 4th axis to LINEAR and its rotary-specific, degree-based settings disappear from the list.

12.4 CONTROLLER Settings (controller.cfg)

Key	Type	Meaning
ui/theme	enum	DARK / LIGHT — overall panel colour theme

ui/accent	enum	TEAL / AMBER / VIOLET / GREEN — accent colour used for highlights
ui/opt_stop	bool	Whether an M1 command in a program is honoured (paused) or ignored
ui/tool_change	enum	IGNORE / PAUSE / MACRO — behaviour on an M6 tool-change command
ui/fixed_feed	mm/min	Forces every feed move to this rate, ignoring the programmed F, whenever set to a non-zero value (section 10.3); 0 disables it and restores normal behaviour
ui/fixed_rapid	mm/min	The same forced-rate behaviour as ui/fixed_feed, applied to rapid moves instead; 0 disables it
ui/jog_vel	mm/min	Continuous jog velocity for the held-arrow keys only (e.g., 2000 mm/min); the encoder wheel's speed instead follows ui/mpg_follow (section 8.1)
ui/mpg_follow	0.02–0.5 s	Wheel follow time — the time constant behind proportional wheel jogging (section 8.1); default 0.10 s. Smaller feels tighter and twitchier, larger feels smoother but laggier
display/lcd_backlight	0–100	Main panel brightness, as a percentage
display/pendant_backlight	0–100	Pendant display brightness, as a percentage (section 20.3)
display/pend_contrast	−1–63	Pendant V2 only. LCD contrast; −1 leaves the pendant's own default in place, which is the safe value since a wrong contrast setting can blank the display entirely
display/pend_ratio	0–7	Pendant V2 only. Contrast resistor ratio; only used when pend_contrast is 0 or higher
display/led_axis_bright	0–100	Brightness of the axis-selection LEDs
display/led_scale_bright	0–100	Brightness of the jog-scale LEDs

CONTROLLER also holds three pages that aren't simple field lists: KEYS, which assigns all six user-assignable keys to an internal function or a macro (section 4.3); CLOCK, which sets the date and time (used by macro logging and licence expiry, section 12.7); and SECURITY, which carries the menu PIN and licence-code entry (section 12.7).

12.5 MACHINE Settings (machine.cfg)

MACHINE settings are grouped into: x, y, z (one per linear axis), axis4, general, homing, homingseq, limits, spindle, and toolzero. Homing keys are covered separately in section 7.2.
Within limits, hard arms the hard-limit kill: the Motion core stops all motion the instant a pin mapped to LIMIT goes active. Enabled by default; E-STOP is entirely separate and never gated by this setting.

The soft setting enables soft travel limits on any referenced axis: a jog stops cleanly at the travel bound (with a SOFT LIMIT message), a program is refused at START if its extent exceeds travel limits, and any single move past travel — from MDI, a macro, or GO ZERO — is refused outright, with the offending line reported. Soft limits are suspended during homing, and an unbounded 4th axis (rollover or continuous) is exempt.
Within spindle: mode selects the control method; min_rpm and max_rpm define the usable speed range (e.g., 0–24,000 RPM); on_delay sets the spin-up dwell before motion is allowed (e.g., 2 seconds); pwm_freq, pwm_min_duty, and pwm_max_duty configure PWM-based spindle speed control (e.g., 1000 Hz, 5%, 95%) for drives expecting that signal instead of 0–10V.

NOTE

The toolzero settings (sensor_height, distance, feed, seek, backoff, pulloff) parametrize the G38 probing cycle — feed (fast approach), seek (slow confirming pass), backoff, distance (travel cap), pulloff (final retract). See section 18.6.

Per linear axis: steps_per_mm (e.g., 1000 steps/mm), max_rate (e.g., 5000 mm/min), accel (e.g., 500 mm/s²), travel_min and travel_max (e.g., −100 to 100 mm), invert, flip_jog, and backlash — these fully describe how the axis moves and how far it can travel.
Within general: step_pulse_us and step_delay_us tune step-pulse timing (e.g., 2 µs and 1 µs); junction_dev (e.g., 0.01 mm) and arc_tol (e.g., 0.002 mm) control path-smoothing tolerance at corners and arcs; smooth_time governs acceleration ramp shaping (below); motion_mode sets the default path mode; block_run is the live single-block toggle from section 10.2.

NOTE

general/smooth_time (seconds, 0 to 0.5, default 0 = off) rounds off the two ends of every speed ramp instead of applying the full acceleration as an instant step, while the middle still runs at full configured acceleration. This protects a stepper motor from stalling at ramp starts or sharp corners, and often lets the configured acceleration be raised safely — net faster overall. It doesn't apply to homing seeks (extra stopping distance there is just overtravel into a switch) or winding-scale 4th-axis moves (entry/exit must cruise smoothly through internal chunk boundaries).

NOTE

general/motion_mode (G64 BLEND / G61 EXACT, default G64 BLEND) is the path mode every program and MDI line starts in. G64 blends corners within junction_dev; G61 stops fully at the end of every block. A G61 or G64 word inside the file wins from that line on — this setting only decides what applies when the file doesn't say. Use G61 EXACT where corner rounding would be a dimensional error (drilling patterns, fine engraving); leave G64 BLEND for ordinary contouring, where G61 is both slower and harder on the machine. Changing this re-parses a loaded program, since the mode is read at parse time. Typing G61 or G64 at MDI updates this setting live (same re-parse) but doesn't save it — saving MACHINE ▸ GENERAL (F2) makes an MDI-typed mode the new power-on default. A G61/G64 word inside a file or macro stays scoped to it and never changes this setting.

NOTE

general/block_run (default off) is the same single-block mode from section 10.2, exposed here as a stored setting. Unlike its neighbours it's also a live mode: a key or mapped input can flip it while the machine runs, and this row just reflects the current state. Saving the page (F2) makes the current state the power-on default.

Within axis4: letter selects the axis designation (A, B, or C); type chooses LINEAR or ROTARY; parallel, steps_per_deg, max_rate, accel, travel_min, travel_max, rollover, continuous, invert, flip_jog,

and backlash configure its motion, same as the linear axes. Rotary-only rows (like rollover) are hidden when the axis is LINEAR, and units switch between degrees and millimetres to match.

12.6 STOCK Settings (stock.cfg)

These settings only affect the 3D toolpath preview on the VIEW screen — no effect on actual machine motion, purely for visualizing a job before it runs.

Key	Meaning
stock/enable	Whether to draw the stock block or cylinder at all in the 3D view
stock/shape	BOX / CYLINDER
stock/mode	AUTO (calculated automatically from the loaded program's extents) / MANUAL
stock/zref	TOP / BOTTOM — which face of the stock Z zero is measured from
stock/margin	Extra clearance added around the AUTO-calculated extents (e.g., 5 mm)
stock/size_x/y/z, origin_x/y/z	Dimensions and origin position for a BOX-shaped stock
stock/diameter, length, position	Dimensions and position for a CYLINDER-shaped stock

12.7 Menu PIN and Licensing

The menu can be protected by a 6-digit PIN, defaulting to 123456. The PIN only gates settings — jogging, homing, running a program, and MDI are never blocked by it, so a forgotten PIN is a service call, not a stopped machine.

> ⚠ **WARNING**
> A forgotten menu PIN can only be reset at a Tiny Controls service centre — record it before changing it from the default. Five wrong attempts lock the PIN prompt for 60 seconds, doubling on each further failure.

CONTROLLER ▸ SECURITY carries both PIN management and licensing:

- CHANGE PIN (F3) — enter the new PIN twice to confirm.
- ENTER CODE (F4) — enter the 12-character licence code. The page shows this controller's DEVICE ID (format XXXX-XXXX-C); send that to Tiny Controls for a code. A new code replaces whatever licence period was active; one reserved code unlocks the controller permanently.

Entry uses OK to confirm, F4 as backspace, F3 to clear the field. Licence codes use only unshifted keypad characters (A F G H L M X Y Z, plus digits), so a code can always be typed without Shift.

> **NOTE**
> Without a valid licence, only CYCLE START is blocked — everything else works normally, so an expired or missing licence never strands a machine mid-setup.

> **NOTE**
> Set the CLOCK (CONTROLLER ▸ CLOCK) before entering a time-limited licence code, since expiry is measured against it. Anti-rollback protection clamps to one day past the active

licence, so winding the clock back can't extend a licence — or permanently poison one. The SECURITY page prints the three numbers the verdict is computed from (today / watermark / expiry, in days) for support calls.

NOTE

"p" firmware — built without the RTC crystal — has no CLOCK group: the licence always reads PERMANENT, and ENTER CODE simply answers that no code is needed. The menu PIN works the same as on any other unit. These units also skip the crystal wait at boot, so they start a few seconds faster than an RTC-equipped board.

12.8 Settings Commit Immediately (Idle-Only Editing)

A setting takes effect the moment it's confirmed, not only once the page is saved — config pages edit the live values directly, and saving (F2) only decides whether those values also reach the SD card. Because of this, editing is refused unless the machine is idle: the panel reports SETTINGS ARE IDLE ONLY - machine is running. That covers a program running or held, a reverse in progress, a macro owning the machine, and the encoder wheel actively turning — jogging drops back to idle the instant the wheel stops.

Never blocked: opening the menu, reading any page, and everything under SYSTEM — DIAGNOSTICS with its live pins and LIMIT OVR (F4), ABOUT, HELP — exactly what an operator needs when something's already gone wrong. The one editable exception is DISPLAY (panel and pendant backlight, LED brightness, pendant contrast): none of it can move an axis, and dimming the panel mid-job is reasonable.

NOTE

This restriction exists because the alternative was worse. Changing steps_per_mm mid-cut would reinterpret the step counters under an axis already moving; a MAPPING polarity bit could flip a LIMIT input's sense mid-job, causing a spurious kill or quietly disarming real protection; changed max_rate, accel, or corner tolerance would leave already-planned blocks running on the old numbers while newly planned ones used the new. An older idle gate that only applied to the save step, not the edit itself, was worse: the edit took effect immediately, the save was refused, and the machine ran on numbers that existed in no file at all, reverting silently on the next reboot.

NOTE

A few settings only take effect at parse time — motion_mode (section 12.5), and the active work coordinate system for a file that names its own G54–G59 (section 6.2). Those re-parse a loaded program whenever they change, since being idle isn't enough when the program in memory was parsed against the old value.

Chapter 13 — I/O Mapping

Every input and output function on the TNC-M34 is assigned to a physical pin from MENU ▸ MAPPING — nothing is wired by default beyond a minimal starting map, giving full flexibility to match whatever wiring already exists. Pins are numbered 0–15 for both inputs and outputs, matching the numbers printed on the hardware and shown on the DIAGNOSTICS screen (Chapter 14).
The SAVED column beside each function name shows the assignment as last saved, not whatever's currently being typed, so a pin can be checked against its old value while retyping. A row tinted amber shares its pin with another row — often intentional, like all four LIMIT functions sharing one pin — but worth a second look. F3 = CLEAR unassigns the pin under the cursor; committing an empty value just cancels the edit rather than writing pin 0 by mistake. Nothing reaches the SD card until F2.

13.1 Input Functions

Function	Notes
HOME X / Y / Z / 4	Homing switch for each axis; may share a physical pin with another function (see section 7.5 and 12.2)
LIMIT X / Y / Z / 4	Hard limit switch; kills motion via the Motion core
PROBE	Drives the G38 probing cycle, and is also readable directly from a macro (see section 18.6)
E-STOP	External emergency-stop chain; a Motion-core kill that is never overridable
CYCLE START	Physical cycle-start button
FEED HOLD	Physical feed-hold button
STOP	Physical program-abort button
MACRO 1…4	General-purpose inputs, readable from macros as #in_MACRO_IN1…4 (e.g. for a guard switch); note that this does not itself launch a macro automatically
BLOCK RUN	Toggles single-block mode (section 10.2) on each rising edge — a mode switch, not a step button; the step input remains CYCLE START. Sharing a pin with START, HOLD, or STOP means this function loses out under the edge-action priority rule below

13.2 Sharing One Input Between Functions

Several functions may legitimately be pointed at the same physical pin — in some cases that's exactly how the system is meant to be wired. But the three broad categories of input behave differently when they share a pin, so it's worth understanding each before wiring a shared switch:
Passive reads share freely, with no downside. PROBE and MACRO 1–4 are simply levels read on demand whenever a macro asks for them, so reading the same pin through more than one costs nothing and causes no conflict.
Masked kills share by design. LIMIT X, LIMIT Y, LIMIT Z, and LIMIT 4 are OR'd into a single mask handed to the Motion core — why the factory default points all four at one pin. A trip on that pin reliably stops the machine, but can't say which axis caused it. E-STOP shares the same OR'd arrangement and always takes priority, since its kill is never overridable.

> ⚠ **WARNING**
> Edge-triggered actions follow a one-press, one-action rule: CYCLE START, FEED HOLD, and STOP each fire on the rising edge of their input. If one pin carries more than one of these, only the most conservative action runs — STOP beats FEED HOLD beats CYCLE START. A button wired to more than one of these always behaves as the safest one, never as a start button. The MAPPING page warns whenever this applies, since every other function on that pin is effectively discarded.

> **NOTE**
> A HOME input also wired as a LIMIT is the common one-switch-per-axis-end arrangement. Because homing deliberately drives into the switch, having the hard-limit kill armed at the same time would stop the machine at the exact moment homing succeeds. To prevent this, homing is refused in that case, naming the axis, until LIMIT OVR is engaged (DIAGNOSTICS F4, or a dedicated panel key, depending on configuration). Hard limits are never silently overridden — they stay armed unless turned off deliberately. With limits/hard disabled, or no LIMIT pin mapped for that axis, the shared pin behaves as an ordinary home switch and homing proceeds normally.

13.3 Output Functions

Function	Notes
SPINDLE EN	Spindle enable output
SPINDLE DIR	Spindle direction (reverse, for M4); unassigned by default
FLOOD	Flood coolant, activated by M8 (stored and scripted internally as COOLANT)
MIST	Mist coolant, activated by M7
MACRO 1…4	General-purpose outputs, driven directly from macros and from M62–M65

13.4 Polarity

Every input has its own invert flag (active-low = a normally-open switch pulled to ground), and every output has one too. All polarity settings live here, in MAPPING — never in the homing settings. The homing invert key only sets the direction of the seek move; it has nothing to do with switch polarity.

13.5 Using I/O from Programs and Macros

- M62 Pn / M63 Pn — set or clear output n, synchronised with motion.
- M64 Pn / M65 Pn — set or clear output n immediately, independent of motion.
- M66 Pn — wait for input n to reach a defined state.
- From a macro: read #in5 or #in_PROBE directly as a variable, or use WAITIN to block until an input reaches a level.

P accepts a pin number from 0 to 15. See Part II for the complete macro-level I/O surface, including named (mapped) pin access.

Chapter 14 — Diagnostics

MENU ▸ SYSTEM ▸ DIAGNOSTICS shows all 32 physical I/O pins at once — 16 inputs, 16 outputs — whether or not they're assigned to a function, with live electrical state shown by colour. This is the screen for verifying wiring before trusting a new mapping in production.

Key	Action
Up/Down, wheel	Select a pin from the list
F1	Force the selected output ON
F3	Force the selected output OFF
F5	ALL OFF — force every output off at once
F2	BACK — return to the previous screen

> **NOTE**
> Outputs forced from this screen are forced for real — they physically energize the wired device, exactly as a program would. Always use ALL OFF before leaving, to avoid leaving an output unexpectedly energized.

Chapter 15 — Tuning

MENU ▸ SYSTEM ▸ TUNING gives a per-axis tuning screen for checking real-world motion quality against actual hardware — useful when commissioning a machine or diagnosing a mechanical issue. Select the axis, choose a test mode, and press ESC to step back out one level at a time.

Chapter 16 — Macros

MENU ▸ MACROS lists every .mac file in 0:/macros, for quick access to run any of them from the panel:

- OK runs the currently selected macro (IDLE only).
- Start resumes a macro waiting at an operator prompt.
- ESC aborts the running macro at any point.

Macros can also be bound directly to the MACRO 1–4 physical inputs, so a machine can launch a routine — a tool-change sequence, say — from a dedicated button rather than the panel menu.

NOTE

Setting the F3 key action to RUN MAC (CONTROLLER ▸ INTERFACE, section 12.4) puts this same macro list directly on the main-screen F3 key, without needing the menu PIN first. Routine work like a tool-zero cycle is then one keypress away, while the menu PIN still guards settings.

NOTE

The full macro language — variables, conditionals, WAITIN, digital I/O access, operator prompts, and tool-handling routines — is documented in Part II.

Chapter 17 — Recovery After Power Loss

If the controller loses power mid-program, the next boot offers a guided recovery instead of forcing a fresh start. Machine state and the loaded program file survive the power loss, held in battery-backed RAM.
The recovery wizard has four labelled steps, and nothing on the machine moves until the operator says so:

Step	What Happens
1/4 REFERENCE	Re-home the machine using its switches, or jog it manually to a known position — the stored global reference point (Chapter 7) is the intended aid for this step
2/4 STATE POSITION	The recovered machine position and the saved program line number are displayed for the operator to confirm
3/4 OUTPUTS	The operator marks which outputs must be switched back on before motion resumes — such as the spindle or coolant — with the spindle spin-up dwell honoured automatically
4/4 CONFIRM	A final review screen, followed by Start to actually resume the program

At the first prompt after a power-loss reboot, OK restores the recovered position and file without starting the program or spindle, leaving the operator in control of the next step; ESC discards the recovery information and starts the controller unreferenced, as if from a normal cold boot.

> **NOTE**
> The saved program line number is shown on screen and can be written down. It’s also exactly the line number RUN FROM LINE (section 10.4) will expect if resuming manually instead.

Chapter 18 — G-code Reference

The TNC-M34 interprets a broad, standards-compatible set of G-codes and M-codes: motion, canned drilling and boring cycles, coordinate system handling, spindle and coolant control, and digital I/O — enough to run programs from most common CAM packages unmodified. This chapter lists every supported code, the smaller set recognised but not implemented, and exactly how each unsupported case is handled, so nothing is ever silently mis-executed.

18.1 Motion and Modal

Code	Meaning
G0 / G1	Rapid positioning move / feed-rate linear move
G2 / G3	Clockwise / counter-clockwise arc interpolation — centre given by I/J/K, or by radius with R (see the NOTE below)
G4 Pn	Dwell (pause) for n seconds
G10 L1 Ph Zv	Write the tool table — set tool length offset h (0–6, the same H number G43 takes) to value v; persists automatically in offsets.cfg
G10 L2 Pn X/Y/Z	Set the origin of work coordinate system n (1–6, corresponding to G54…G59)
G17 / G18 / G19	Select the XY / XZ / YZ plane for arc interpolation
G20 / G21	Set units to inches / millimetres
G28, G30	Rapid move to the predefined reference position
G28.1, G30.1	Store the current position as the reference position (kept in RAM)
G40	Cancel cutter compensation (the only supported state — see section 18.4)
G43 Hn / G49	Apply / cancel tool length offset h to the Z axis
G52	Apply a temporary local coordinate offset
G53	A single, non-modal move expressed directly in machine coordinates
G54…G59	Select work coordinate system 1 through 6
G61 / G64	Exact-stop mode / continuous (blended) path mode
G90 / G91	Absolute distance mode / incremental distance mode
G90.1 / G91.1	Arc centres given as absolute coordinates / incremental offsets
G92 / G92.1	Set a temporary coordinate system offset / clear it
G93 / G94	Inverse-time feed mode / standard units-per-minute feed mode
G98 / G99	Canned cycle retract to the initial Z height / retract only to the R plane

NOTE
For a G2/G3 arc given by radius (R) rather than centre (I/J/K): a positive R takes the short

way round (180° or less), a negative R the long way round. An R form whose start and end points coincide cuts a full circle, centre placed R along the active plane's first axis — a call the controller has to make on the machine's behalf, since R alone leaves the true centre ambiguous; use I/J/K when the exact centre matters. An R too small for the actual chord distance (with a small 0.1% rounding tolerance, read as an exact half circle) refuses the whole file, naming the line. Centre words that contradict the active plane are corrected automatically when unambiguous — I and K together imply G18, J and K imply G19, and the console notes the correction — but a centre word that belongs to no plane (a lone K under G17, say) refuses the file rather than guess.

18.2 Canned Cycles

For every canned cycle below, R sets the retract plane, Q the peck increment, and P the dwell time in seconds where applicable. G98 (default) returns the tool to its initial Z height between operations; G99 returns it only to the R plane for faster cycles. G80 cancels whichever cycle is active.
Fully implemented, including spindle handling where relevant:

Code	Meaning
G73	Peck drilling cycle with partial (chip-breaking) retract between pecks
G81	Simple drilling cycle
G82	Drilling cycle with a dwell at full depth
G83	Peck drilling cycle with a full retract to the R plane between every peck
G85	Boring cycle, feeding in and feeding back out
G89	Boring cycle with a dwell at depth, then feeding back out

Implemented for motion only, spindle handling left to the operator or the program:

Code	Meaning	Gap
G84	Tapping cycle	Does not reverse the spindle direction on the retract stroke
G86	Boring cycle	Does not stop the spindle at full depth
G88	Boring cycle	Does not stop the spindle at full depth

⚠ **WARNING**
These three run their motion correctly but deliberately leave spindle control to the operator or the program. Spindle direction output isn't yet hardware-verified, and there's no spindle encoder, so true encoder-synchronised tapping isn't physically possible — G84 relies on the operator setting F = pitch × RPM by hand. Every affected line prints an on-screen warning naming itself when it runs. Treat G84 especially as an assisted manual operation, not a fully automatic one, and supervise it.

Not supported at all — a program using one of these is refused at load time, before any motion occurs:

Code	Meaning
G74	Left-hand tapping cycle

G76	Fine boring cycle
G87	Back boring cycle

NOTE
These codes are refused outright, not just skipped, because a canned cycle is modal: any bare X/Y line that follows it is another point of that same cycle, not an independent move. Skipping only the line that defines the cycle would silently omit every subsequent hole or bore while still running the moves between them — far more dangerous than refusing to load. The panel reports both the offending code and its line number.

18.3 M Codes

Code	Meaning
M0 / M1	Program stop / optional program stop (see ui/opt_stop, section 12.4)
M2 / M30	Program end
M3 / M4 / M5	Spindle on clockwise / on counter-clockwise / off (an S word sets the target RPM)
M6	Tool change — the exact behaviour is set by ui/tool_change (section 12.4)
M7 / M8 / M9	Mist coolant on / flood coolant on / all coolant off
M48 / M49	Enable / disable feed and speed overrides
M62 / M63	Set / clear output P, synchronised precisely with motion
M64 / M65	Set / clear output P immediately, independent of motion
M66 Pn	Wait for input n to reach a defined state before continuing

NOTE
A lone S word, with no accompanying M3 or M4, simply retargets the RPM of a spindle already running.

18.4 Not Supported

G-codes and M-codes the controller doesn't implement are handled in three tiers, chosen by how much damage the code could cause if it were quietly ignored during a running program.
Tier 1 — the program is refused at load time. These are all modal codes: once active, they change how every following line is interpreted, so there's no way to offer a partially-correct run. The panel names both the offending code and its line number.

Code	Meaning	Why Ignoring It Would Be Unsafe
G16	Polar coordinates	X and Y would be silently reinterpreted as radius and angle on every subsequent line
G51	Scaling	every subsequent coordinate would be scaled without the operator necessarily expecting it

G68	Coordinate rotation	the entire coordinate frame would be rotated, shifting every subsequent move
G41 / G42	Cutter radius compensation	every toolpath would be offset by a tool radius; running the same path on the centreline instead would gouge material the program expected to leave untouched
G74 / G76 / G87	Left-hand tap, fine bore, back bore canned cycles	these cycles are modal — see section 18.2 for the full explanation
G95	Feed per revolution	the F word changes meaning entirely; interpreted as mm/min instead, it produces roughly a 1000× error, so the program simply would not cut correctly

Tier 2 — accepted and silently ignored, since these are genuinely harmless. Each is the cancel form of a tier-1 code above, and since the corresponding mode was never enabled, there's nothing for the cancel to do: G15 (polar off), G40 (cutter compensation off), G50 (scaling off), G69 (rotation off).
Tier 3 — the individual line is skipped; the rest of the program runs normally:

Code	Notes
G38.2–G38.5	Probing codes — supported only from MDI or the macro PROBE verb, not from inside a loaded program (see section 18.6)
An unrecognised G word with coordinates on the same line	The controller cannot know in advance whether that unrecognised code was meant to repurpose those coordinates — G16, G51, and G68 would all look identical to the parser up to this point, so the safer choice is to skip the line rather than guess
An unsupported M code	Only that specific action is skipped, and it is named on screen; any motion words present on the same line still execute normally

NOTE
An unrecognised G word entirely alone on its own line, no coordinates attached, is silently ignored — there's nothing on that line for it to have reinterpreted.

⚠ **WARNING**
M13 and M14 (spindle on plus flood coolant on, clockwise and counter-clockwise) are fully supported — internally they decompose exactly into M3 or M4 plus M8. They're called out here because failing to support them would be the dangerous case: a program using them would believe the spindle was already running and feed a stationary tool straight into the workpiece.

⚠ **WARNING**
An axis word this machine can't consume gates the program rather than being silently dropped. This covers a rotary letter that isn't the configured 4th axis (a B word while the 4th axis is set to A, say), any A/B/C word when the 4th axis is disabled entirely, and U/V/W/E — none of which are axes on this controller (on lathe posts U/W are normally incremental X/Z,

and E collides with exponent-notation numbers, which this parser doesn't read). Such a word used to just vanish while the rest of its line still ran — partial motion, or a whole line the program believed had executed when nothing moved, leaving the tool in the wrong orientation. The file now still loads, so VIEW can show what's wrong, but with an on-screen warning naming the letter, the first line it appears on, and how many lines are affected (e.g., USES B - 4TH AXIS IS A (LINE 12, x47)) — and Start and RUN FROM LINE both refuse to run it. Not overridable, because the motion plan in memory is genuinely missing motion the file asked for; the fix is the post-processor, or the 4th-axis configuration in MACHINE ▸ 4-AXIS, whichever is actually wrong. From MDI, the line is simply refused on the spot.

18.5 4th Axis

The 4th-axis letter is fully configurable — A, B, or C — via the axis4 setting (section 12.5). A rotary 4th axis is always in degrees and unaffected by G20/G21; a linear 4th axis scales with the other linear axes as expected.

NOTE

Very long moves are expected on the 4th axis: winding-type work commonly drives it through large cumulative values, LINEAR or ROTARY. Such moves are chunked internally to keep the step arithmetic exact over long distances, with the chunk size set large enough that a normal-sized program never reaches the boundary.

NOTE

With rollover enabled, a rotary axis's display wraps to 0–360 degrees for readability. The internal position isn't reset mid-program when this happens, so a program winding through many full turns keeps tracking correctly.

18.6 Probing — G38

A probing move is fundamentally different from an ordinary move: its endpoint is unknown in advance, since it depends on exactly when the probe input trips, and the cycle retracts and re-approaches on its own for an accurate reading. Because of this, probing is available from MDI and the macro PROBE verb, but deliberately not from inside a loaded program — a G38 line in a loaded file is skipped, and the event reported on screen (see also Chapter 21, Known Gaps).

Code	Direction	On No Contact
G38.2	Toward the target, stop on contact	Reports an error and retracts
G38.3	Toward the target, stop on contact	No error is raised (check #probe_hit instead)
G38.4	Away from the target, stop when contact is lost	Reports an error and retracts
G38.5	Away from the target, stop when contact is lost	No error is raised

The command takes the form G38.2 Z-30 — a single axis only, and the number is a relative distance, not an absolute coordinate. Any F word on the line is ignored: probing speeds come from CONFIG → MACHINE → TOOL ZERO (feed for the fast approach, seek for the slow confirming pass, backoff, distance as the travel cap, pulloff for the final retract), since those six fields parametrize this cycle.

NOTE

The cycle always runs a fast approach, retracts by the configured backoff distance, then re-probes slowly for an accurate reading — the reported position is always the capture from that final, slow pass. The trip is detected, the position captured, and motion stopped entirely on the Motion core, inside the step interrupt, so measurement accuracy never depends on the UI frame rate.

NOTE

The reported result is a genuine machine coordinate, captured the instant the input fires, and matches exactly what the DRO shows at the moment of contact — including on an axis with invert set or backlash configured.

Both retract moves — backoff and pulloff — are clamped to the axis's own travel limits, and an upward Z retract is additionally clamped to the same ceiling every safe-Z move respects (one sensor-clear margin below an upward-seeking home switch, section 7.3). So a generous 50 mm pulloff near the top of Z travel simply stops at the travel bound instead of driving the tool back into the switch that referenced the axis. If the backoff retract fails to reopen the probe, the cycle stops and reports PROBE: STILL TRIGGERED AFTER BACKOFF rather than re-probing from an already-closed probe — raise the configured backoff distance if this appears.

NOTE

STOP or ESC ends a probing cycle cleanly at any point, during either pass, whether started from MDI or a macro.

NOTE

A probing move is refused before any motion if: the axis isn't referenced, no PROBE pin is mapped, the probe is already showing tripped, or the PROBE pin is shared with a LIMIT function while hard limits are armed (engage LIMIT OVR first).

Results are readable from a macro afterward as #probe_hit and #probe_x/y/z/a (machine coordinates), and can be stored with TOOLLEN (tool offset table) or WORKZERO (active work coordinate system). See toolzero.mac and workzero.mac, provided by default in the macro folder, and Part II, section 10, for worked examples.

Chapter 19 — Alarms and Messages

Status and alarm messages appear in the top bar. These are the ones most worth knowing, particularly during homing, since they cover most real-world commissioning issues:

Message	Meaning
HOME <axis> FAILED: switch not seen	The seek move ran the full length of the travel span without ever tripping the switch — usually caused by a wrong seek direction, or a switch that is not actually wired up
HOME <axis> FAILED: switch will not release	The axis backed off as expected, but the switch remained tripped — check the wiring, the polarity setting, or whether the switch itself is mechanically stuck
HOME <axis> FAILED: switch re-made during clearance	The switch closed again while the axis was backing away from it — usually caused by contact bounce or a marginal, noisy sensor
HOME <axis>: park is behind the switch	The configured park coordinate would drive the axis back onto the switch it was just referenced from; the axis is still correctly referenced, but the park move itself is refused for safety
HOME <axis>: park <n> too far from switch	The configured park coordinate is further from the reference position than the axis is physically able to travel — almost always the result of a typo in configuration
HOME <axis>: park short by <n>	The park move did not reach its full configured coordinate, typically due to a travel limit; the reference position itself, however, remains valid
HOME <axis> FAILED: did not stop after release	The axis never came fully to a standstill after backing off the switch
HOME <axis> FAILED: clearance move did not finish	The clearance (back-off) move never completed
HOME <axis> FAILED: park move did not finish	The move to the park coordinate never completed
HOMING DONE - <axes> NOT referenced	One or more named axes failed or were skipped during the cycle; every other axis referenced successfully
HOMING: switch already made - releasing	Normal, expected behaviour — the pre-check found the switch already closed and is clearing it before seeking
HOMING: IDLE only	Homing was requested while the machine was already busy running something else
HOMING: Z to safe height	Normal, expected behaviour — this is the Z-

	safe lift performed automatically before homing X and Y
SAFE Z (work <n>) capped - using <n>	The work-frame safe-Z height converts to a machine height beyond machine zero, Z travel, or the Z home switch position, so the lift was automatically clamped to a safe value (section 7.3)
ABORT: no retract - Z <n> is above safe <n>	The abort retract was skipped because Z was already at or above the safe-Z height when Stop was pressed — the retract had nothing to do, rather than being switched off
GO ZERO (no lift - Z already above safe Z)	The GO ZERO lift was skipped for the same reason — Z was already at or above the safe-Z height

NOTE

A single failed axis doesn’t abort the whole homing cycle; every remaining axis still gets its turn, and the final summary names whichever axes ended up unreferenced.

⚠ WARNING

The three "did not finish/stop" messages are watchdog failures: an internal phase waited a full two minutes for the axis to come to rest and gave up. These should never occur normally. If one does, treat it as a genuine fault worth investigating, not a setting to tune.

NOTE

If homing ever appears to hang, press Stop. This aborts the cycle cleanly and releases the lock homing otherwise holds over jogging, MDI, and program start.

Chapter 20 — Firmware Update and Pendant Hardware

This chapter covers updating the controller, updating an attached MPG pendant, and the hardware differences between the two pendant generations the TNC-M34 supports.

20.1 Controller Firmware

The controller firmware updates in the field from an ordinary FAT-formatted USB stick, with no special tools, no PC software, and no key sequence to remember:

1. Copy M34FWUPD.TNC to the root of the USB stick — not inside a folder, and do not rename it.
2. Plug the stick into the controller's USB-HOST port.
3. Switch the machine on. That is all there is to it.

The controller looks for the file for a few seconds after every power-on. If it isn't there, the machine boots normally, so leaving a stick in the slot permanently does no harm.
The eight panel LEDs act as the progress display during an update, filling upward one stage at a time. The bottom four are the SCALE column, the top four the AXIS column:

LED (bottom to top)	Stage
x1000	Stick found and mounted
x100	M34FWUPD.TNC opened
x10	Header accepted — correct product and board
x1	Signature verified (the longest step, a few seconds)
A	Erasing
Z	Writing the main processor
Y	Writing the motion processor
X	Reading it all back and checking

The blinking LED is the stage running now; every LED below it stays solid. When the run ends, all eight LEDs together tell you the outcome:

All eight LEDs	Meaning
Blinking together	Update finished — the machine resets itself
Solid	This firmware was already installed; booting normally
One LED blinking fast	Stopped at that stage

> **NOTE**
> Nothing is erased until the signature passes. A file that is corrupted, truncated, or meant for a different product or board revision is refused before any writing begins, leaving the existing firmware untouched — the panel shows the failure briefly and then boots what was already there. A wrong stick cannot take a machine out of service.

If an update is interrupted — a power cut mid-write, for example — the controller knows. On the next power-on it refuses to start a half-written firmware and waits for the stick instead, with the bottom LED blinking. Plug the stick back in and it finishes the job; no state you can reach by pulling the plug needs a service visit.

To force a reinstall, hold Rapid-Override while switching on. The LED bar flashes twice to confirm, and the firmware is rewritten even though it already matches the version on the stick. Use it when a unit is behaving oddly and you want a known-good image back on it.

> **NOTE**
> After an update, review any configuration file whose menu version changed — the panel migrates these automatically, but confirming a migrated homing coordinate by eye (section 7.2) is worth doing before relying on it in production.

> **NOTE**
> Older controllers shipped with a different bootloader that expects a file named TNCM34P1.bin and requires Rapid-Override, then OK, at power-on. If the panel shows no LED progress bar during an update, it has that bootloader and needs the older file. The two are not interchangeable.

20.2 Pendant Firmware Update (V2 Only)

A Pendant V2 updates over the same RS-485 cable used for normal operation, from MENU ▸ SYSTEM ▸ PENDANT UPDATE. The pendant firmware image travels bundled inside the controller's own firmware, so updating the controller always delivers matching pendant firmware — nothing separate to copy.
The PENDANT UPDATE page shows link state, the pendant's reported firmware version, and the version bundled in the controller. F3 starts the update, which runs through six phases:

Phase	What Is Happening
ENTERING BOOTLOADER	The pendant is asked to reboot into its bootloader
ERASING	The staging slot is cleared (up to 5 seconds)
WRITING	The firmware image is streamed across, with a progress bar shown
VERIFYING	The pendant checksums the image it received
PROMOTING	The image is copied into the execution slot (up to 10 seconds)
CONFIRMING	The new version is read back over the link to confirm success

The whole process takes around 20 seconds. The machine must be IDLE to start, since the pendant reboots partway through and its display goes blank for part of that time.

> **NOTE**
> It's safe to interrupt this update at almost any point. Everything before PROMOTING writes only to a spare staging slot, so a failure or power cut leaves the pendant running its existing firmware — just retry from the same page. If power is lost during PROMOTING itself, the pendant's own bootloader finishes the job automatically next power-up, no operator action needed. If the page reports the pendant sitting in its bootloader with no transfer in progress, press F3 to re-send.

Re-sending the same version already on the pendant is allowed — the update is always operator-initiated, so an explicit re-flash or downgrade is a legitimate choice, not a mistake to guard against. The version readout won't change in that case; the pendant's behaviour afterward is the confirmation the re-send happened.

A Pendant V1 is no longer detected at all: it answers no probe on the link, so the page reads NO PENDANT ON THE LINK rather than offering an update it can't perform. V1 units are converted to V2 at a Tiny Controls service centre.

20.3 Pendant Hardware

Pendant V2 is the only pendant generation this firmware speaks to. V1 support was retired once every shipped V1 unit was returned and reflashed to V2 — the controller no longer speaks the V1 protocol, so a V1 pendant left connected simply answers no probe and is treated as if nothing were connected. A V2 pendant is detected automatically — nothing to configure.

	Pendant V2
Display	Partial-region refresh, giving a noticeably fast-updating DRO
Contrast, from the menu	Available (display/pend_contrast, section 12.4)
Backlight, from the menu	Available
Long-press keys	Supported
Firmware update over the cable	Supported (section 20.2)

At power-on, the controller probes for a pendant three times in quick succession, then keeps probing every 2 seconds — a pendant plugged in after power-up is still picked up within a couple of seconds, and a machine with none connected pays essentially no cost for the ongoing probing.

> **NOTE**
> "No pendant" isn't treated as a fault. With nothing on the RS-485 link, ABOUT reads PENDANT: NONE (ENCODER-ONLY IS OK) and the PENDANT screen says NO PENDANT - OR ENCODER-ONLY. The panel's own built-in MPG encoder wheel is wired directly to the controller's MCU, not through this link, so it keeps working exactly as before — no link fault to chase. A V1 pendant left connected reads the same way, since it simply doesn't answer the probe.

The pendant keys mirror the main panel: jog step, XY axis, Z axis, jog axis, start/pause, and a spindle toggle (toggling M3/M5, gated to machine-idle like MDI). If a V2 display ever shows mixed or stale content — say after its own disconnect screen raced a reconnect — it self-corrects within about 10 seconds, since the controller periodically re-sends the whole frame precisely because the pendant sometimes draws its own screens independently. MENU ▸ SYSTEM ▸ PENDANT UPDATE and MENU ▸ SYSTEM ▸ ABOUT both report whether a pendant is currently found.

Chapter 21 — Known Gaps

These limitations are stated plainly and up front, so no operator discovers one unexpectedly mid-workpiece:

- G84, G86, and G88 run their motion correctly but leave the spindle exactly as it was — G84 in particular won't reverse the spindle to retract from a tap. Spindle direction output isn't yet hardware-verified, and there's no spindle encoder, so true encoder-synchronised tapping isn't possible on this hardware. Treat these three as assisted manual operations (section 18.2).
- G38 probing is available only from MDI or a macro, never from inside a running program — a G38 line in a loaded file is skipped and reported rather than executed (section 18.6).
- Cutter compensation, coordinate scaling, coordinate rotation, polar coordinates, and feed-per-revolution aren't supported. A program using any of them is refused at load, naming the code and line, since each changes how every following line would be interpreted (section 18.4).
- G74, G76, and G87 canned cycles aren't supported, and are refused at load the same way (section 18.2).
- The spindle/mode setting (0–10V vs. PWM control signal) has no effect — only a PWM output exists on this hardware, so the setting is inert regardless of configuration.

PART II

Macro Programming

Chapter 22 — Macro Programming Overview

Macros are short, operator-facing scripts that drive the machine's own motion and I/O primitives, with a bit of programming logic on top — essentially, scriptable MDI enriched with variables, loops, and operator prompts. They're the recommended way to automate tool changes, probing routines, parking sequences, safety interlocks, and automatic tool changers, without a separate PC or CAM software.
A macro is not a G-code program. Rather than describing a toolpath, it issues individual moves and M-codes one statement at a time, and can pause for operator input, branch on a condition, loop, wait on a physical input, and talk to the operator in between — capabilities a plain G-code program doesn't have.

22.1 Where Macros Live and How to Run Them

- Macros are plain text files with a .mac extension, in 0:/macros/ on the SD card. Edit them on a PC and copy across, or edit directly on the panel by opening the file from FILES — like any file outside 0:/gcode, a macro always opens view-only/edit, not as something runnable from that screen.
- Every macro in the folder appears automatically in MENU → MACROS; highlight the one you want and press OK to run it.
- A macro named toolchange.mac runs automatically on an M6 line, provided CONTROLLER CONFIG → tool_change is set to MACRO. On that call, the requested and current tool numbers are available as #treq and #tcur.
- Macros can also be bound directly to a physical key (section 4.3), so a probing routine or park sequence is one press away, no menu needed.

NOTE
The firmware provisions its bundled macros onto the card at boot, write-if-missing: an edited copy of a shipped macro is never overwritten, and deleting a bundled macro just makes it reappear next boot. To remove a shipped macro for good, open it and delete the END statement, leaving the file otherwise empty, rather than deleting the file.

NOTE
While a macro runs, it owns the machine: jogging, MDI, and starting a program are locked out. At a PROMPT or CONFIRM pause, the operator can jog — positioning for a manual step is the point of that pause — and the macro resumes with CYCLE START. Changing a reference position, the active WCS, or homing status stays blocked throughout, since any of those would move the coordinates the macro is currently working from.

Chapter 23 — Macro Program Structure

A macro file follows a simple, line-oriented structure: one statement per line, each falling into one of four categories:

Form	Meaning
VERB args	A command, covered in detail in Chapters 23 through 26
:name	A label, used as a jump target for GOTO or IF
; text	A comment, running to the end of the line
(text)	A whole-line comment, but only when the opening parenthesis is the very first character on the line

> **NOTE**
> Comments use a semicolon (;), never parentheses. This is a deliberate difference from G-code: in the macro language, ( ) is expression grouping, so MPOS X[(#a-1)*2] is genuine arithmetic, not a comment. A leading (at the very start of a line is tolerated as a whole-line comment, for operators used to G-code conventions, but ( ) should never be used mid-line for anything other than grouping an expression.

A macro ends at an explicit END statement or at end of file, whichever comes first. No overall length limit, though it’s capped at 48 distinct variables per macro.

Chapter 24 — Motion Verbs

Every motion verb below blocks the macro until the move finishes — no queuing several moves at once from within a macro. All motion is in millimetres regardless of G20/G21, and coordinates are X, Y, Z, A words, which may contain variables and bracketed [expressions] (Chapters 27 and 28).

Verb	Emits	Frame	Use
RAPID X.. Y.. Z..	G0	Absolute, work	A fast positioning move in the currently active work coordinate system
MOVE X.. F..	G1	Absolute, work	A feed-rate move — F is required on the first MOVE/MOVER of a macro; see the WARNING below
MPOS X.. Y.. Z..	G53 G0	Absolute, machine	A move to a fixed physical machine position — ideal for tool-change spots or parking
RAPIDR X.. Y..	G91 G0	Relative	A rapid move expressed as a delta from the current position
MOVER X.. F..	G91 G1	Relative	A feed-rate move expressed as a delta from the current position

NOTE

Use MPOS for anything that must land in the same physical location regardless of the active work coordinate system — tool-change positions, park corners, automatic tool-changer pockets.

```
MPOS  Z0                  ; retract to machine top (safe)
MPOS  X0 Y0               ; change position, machine coords
RAPID X50 Y25             ; rapid in the active WCS
MOVE  Z-5 F300            ; feed down at 300 mm/min
MPOS  X[#x0 + (#treq-1)*#pitch] Y[#forky]    ; computed pocket
```

⚠ WARNING

A macro starts with a clean modal state every run, and F only stays modal within that run — so a MOVE or MOVER with no F in effect aborts the macro outright with UNSUPPORTED G1 WITHOUT F, rather than running at the axis's full, unclamped speed, which is what an F=0 feed move actually did previously.

⚠ WARNING

The #tz_zsafe trap. homing/z_safe is readable inside a macro as #tz_zsafe, but never write MPOS Z[#tz_zsafe]: that setting is a work-frame height (section 7.3), while MPOS takes machine coordinates directly, and only the firmware's own safe-Z legs do the frame conversion together with the machine-zero, travel, and switch clamps. The shipped macros instead use a plain MPOS Z0 (machine top) for retracts — adapt that line to a specific machine, or use the panel's own GO REF / GO ZERO (bound to a key if convenient, section 4.3) for any travel that genuinely needs a proper, clamped safe-Z lift.

Chapter 25 — Spindle, Coolant and Digital Outputs

The verbs in this chapter take effect immediately — unlike Chapter 24's motion verbs, they're not synchronised with the motion queue.

Verb	Emits	Notes
SPINDLE CW [S<rpm>]	M3 [S..]	Spindle on, clockwise rotation; an RPM value is optional
SPINDLE CCW [S<rpm>]	M4 [S..]	Spindle on, counter-clockwise rotation
SPINDLE OFF	M5	Spindle stop
COOLANT ON / COOLANT OFF	M8 / M9	Flood coolant on / off
MIST ON / MIST OFF	M7 / M9	Mist coolant on / off
OUT <name> ON / OUT <name> OFF	M64 Pn / M65 Pn	Set a general-purpose digital output on or off

25.1 Naming Pins

Pin numbers are zero-based, using the same numbering for inputs and outputs — IN0 through IN15, OUT0 through OUT15 — matching the numbers printed on the board and shown on SYSTEM → DIAGNOSTICS (Chapter 14).

The <name> placeholder can be a mapped function name from MENU → I/O MAPPING (MACRO_OUT1, FLOOD, SPINDLE_EN) or a raw pin reference like OUT7. The flood coolant output is labelled FLOOD on the panel but still answers to its original internal name, COOLANT — both reach the same relay.

NOTE

Prefer mapped function names over raw pin numbers. A mapped name survives future rewiring; a bare pin number has to be found and edited in every macro that used it. Names are matched case-insensitively and ignore spaces/underscores, so MACRO_OUT1, macro out 1, and MacroOut1 are the same function.

CAUTION

The older O<n> pin form (where O1 meant pin 0) has been removed outright, not renumbered to match the new scheme — renumbering it would have let a line like OUT O2 ON go on parsing without complaint while silently switching a completely different relay than the one the macro's author intended. A macro still using O<n> now fails at load time with an "unknown output" error instead. Rewrite it as OUT<n-1>, or better still, switch to the mapped function name for that output.

```
SPINDLE OFF
COOLANT OFF
OUT MACRO_OUT1 ON        ; drawbar release solenoid
OUT OUT2 OFF             ; raw pin, when nothing is mapped to it
```

Chapter 26 — Timing and Inputs

These two verbs cover deliberate pauses and waiting on a physical input from within a macro:

Verb	Meaning
DWELL <seconds>	Pause execution for the given number of seconds (e.g., for spindle spin-down, or letting a valve settle)
WAITIN <name> HIGH|LOW [T<sec>]	Block until the named input reaches the specified level

The <name> placeholder follows the same rule as for outputs (section 25.1): a mapped input function name, or a raw IN<n> pin reference. The optional T<sec> sets a timeout, default 10 seconds; if reached without the input changing state, the macro aborts cleanly with an on-screen message rather than blocking forever.

```
DWELL   1.0                     ; let the spindle wind down
WAITIN  PROBE HIGH T20          ; mapped name, 20 s limit
WAITIN  MACRO_IN1 LOW           ; guard switch, default 10 s
```

To test an input without blocking macro execution, read it as a variable instead (Chapter 28) and branch on it with IF.

Chapter 27 — Operator Interaction and Flow Control

These verbs cover talking to the operator during a macro, and controlling the order statements execute:

Verb	Meaning
MSG "text"	Display a transient status-line message, with no pause
PROMPT "text"	Pause execution; CYCLE START continues
CONFIRM "text"	Pause execution; CYCLE START proceeds, while ESC or STOP aborts the macro instead
ALARM "text"	Abort the macro immediately, with this text given as the reason
SET #var = <expr>	Assign the result of an expression to a variable
IF <expr> GOTO :label	Jump to the given label whenever the expression evaluates to non-zero (true)
GOTO :label	Jump to the given label unconditionally
END	Finish the macro successfully

Any message-producing verb can embed live variable values in its text with %name (Chapter 29 has the full substitution rules) — useful for showing the operator exactly what tool or position is involved:

```
MSG     "Tool change: T%tcur -> T%treq"
PROMPT  "Fit tool T%treq, then CYCLE START"
CONFIRM "Guard OPEN - close it, CYCLE START to retry (ESC aborts)"
IF      #tries < 3 GOTO :check
ALARM   "Guard still open after 3 tries"
```

Chapter 28 — Variables

Every variable in the macro language is written as #name, case-insensitive throughout. These system variables are provided by the controller and are read-only:

Name	Value
#treq / #tcur / #tool	Requested / current tool number; #tool is a read-only alias of #tcur
#x #y #z #a	The live machine position of each respective axis
#in0 … #in15	The current level (0 or 1) of input pin n
#in_<NAME>	The current level of an input, read by its mapped function name — e.g., #in_PROBE, #in_MACRO_IN1
#probe_hit	1 if the most recent probe actually touched, else 0
#probe_x #probe_y #probe_z #probe_a	Machine position captured at the trip of the most recent probe on that axis
#ref_valid, #ref_x #ref_y #ref_z #ref_a	The stored global reference point (section 7.6)
#tz_sensor #tz_dist #tz_pulloff #tz_zsafe #tz_fixed #tz_x #tz_y	The live TOOL ZERO settings (section 12.5), for use in a probing macro's own arithmetic
#tlo	The tool length offset currently in force — the value G43 is applying (0 if G49 is active). Read from the operator's own context, not the macro's: the parser adds this to every commanded Z while the DRO's work reading does not (section 6.1), so a reference position probed with an offset still in force makes the program cut by that same offset. workzero.mac refuses to run rather than guess in that situation
#tool_ofs	The tool table entry for #tool — the length a G43 H<tool> would apply, whether or not that offset is currently active

NOTE
#in_<NAME> is the reading counterpart to WAITIN's named-input form (Chapter 26): it lets a macro branch on an input's state without ever referencing it by raw pin number, keeping the macro portable across wiring changes.

NOTE
Every system variable above is read-only, with two exceptions: SET #tcur = ... and SET #treq = ... are permitted, and write the engine's real tool registers directly — how toolchange.mac records a completed swap. Every other system variable refuses a SET at parse time, so a macro can never fake a probe result or silently rewrite the machine's configuration as a side effect.

NOTE
Probe results persist after the macro that took them has ended, so a follow-up macro — or

> the next MDI check — can still read back #probe_z and the rest.

Any other variable name is a user variable: created on first use, holding a float defaulting to 0, capped at 48 distinct user variables per macro.

```
SET #x0    = 100        ; X of pocket 1
SET #pitch = 40         ; pocket spacing
SET #n     = #n + 1     ; counter
SET #tcur  = #treq      ; operator confirmed the new tool
```

Chapter 29 — Expressions and Substitution

The macro language supports floating-point arithmetic with familiar, C-like operator precedence: + - * / for basic arithmetic; unary - and ! (logical not); comparisons < > <= >= == !=; logical & (and) and | (or). Parentheses ( ) may be used freely for grouping. There are deliberately no built-in functions — no sin, no sqrt — so keep expressions to straightforward arithmetic.
Expressions may appear in a SET statement, an IF condition, or inside square brackets [ ] on a motion line:

```
MPOS X[#x0 + (#treq - 1) * #pitch] Y[#forky]
IF   #treq > 6 GOTO :badtool
SET  #z = #clearz - (#tcur * 2)
```

There are two kinds of substitution used throughout the macro language:

- In move coordinates, a bracketed [expr] evaluates to a number, while a bare #var is simply inlined as its current value.
- In operator-facing messages (MSG, PROMPT, CONFIRM, ALARM), %name is replaced by that variable's current value — a plain integer when whole, otherwise 3 decimal places. Example: MSG "at X%x depth %z".

NOTE

For PROBE, TOOLLEN, and WORKZERO specifically — which evaluate their arguments directly rather than through the text-substitution used for motion lines — [expr] and (expr) are interchangeable, so PROBE Z-[#tz_zsafe] and PROBE Z-#tz_zsafe are exactly equivalent. Arguments to these verbs are evaluated when the statement runs, not when the file is read. This matters for TOOLLEN #tcur = ..., since #tcur is routinely reassigned earlier in the same macro — evaluating at execution time is what files the measurement against the tool genuinely fitted at that point, not whichever tool was loaded when the file was first opened.

Chapter 30 — Execution Model and Errors

Understanding how a macro executes, and how it fails, makes it much easier to write macros that behave predictably in every situation, including errors:

- Statements execute strictly top to bottom. Motion verbs, DWELL, and WAITIN all block until they complete; PROMPT and CONFIRM block until the operator responds.
- A parse error — an unknown verb, a malformed SET, a missing label, or an unrecognised pin or function name — prevents the macro from starting at all, and reports the offending line. Nothing moves, since every parse error is caught at load time, never partway through execution.
- An ALARM statement, a WAITIN timeout, a CONFIRM cancelled by the operator, or pressing STOP, all abort the macro cleanly and return the machine to idle.
- As the M6 tool-change handler, a macro reaching END hands control back to the controller: spindle and coolant states are restored automatically, and the loaded program resumes at the tool-change line, using a safe-Z approach move.

NOTE

SYSTEM → DIAGNOSTICS (Chapter 14) lists every input and output pin with its live electrical state and mapped function name — the fastest way to confirm a WAITIN statement is watching the pin you intended, and to force an output on or off by hand before trusting a new macro with it in production.

Chapter 31 — Probing in Macros

PROBE <axis><dist> [NOERR] [AWAY] runs the firmware's complete G38 probing cycle in a single statement: a fast approach at the configured toolzero/feed rate, a retract by the configured backoff distance, a slow confirming pass at toolzero/seek, and a pulloff retract once contact is confirmed. The trip decision, position capture, and motion stop all happen entirely on the Motion core, inside the step interrupt, so measurement accuracy never depends on the UI frame's responsiveness.

```
; workzero.mac - set work Z0 on a surface through a touch plate
MSG     "WORK ZERO: probing Z"
IF      #in_PROBE == 1 GOTO :made      ; already touching? do not plunge
SPINDLE OFF
COOLANT OFF
DWELL   0.5
PROMPT  "Plate on the work, tool above it? CYCLE START"
PROBE   Z-#tz_dist                     ; the whole cycle, one statement
WORKZERO Z = #probe_z - #tz_sensor     ; plate thickness comes off here
MSG     "Work Z0 set (plate %tz_sensor)"
GOTO    :done
:made
ALARM   "PROBE already made - clear it before work zero"
:done
END
```

> ⚠ **WARNING**
> Never build a manual touch-off out of MOVE combined with WAITIN — it's the wrong tool for the job, for two reasons. First, WAITIN is only evaluated once per UI render frame, so the tool keeps travelling between polling checks, and the motion pipeline carries a backlog on top — the distance ultimately recorded reflects the stopping distance of the move as much as the actual surface. Second, this approach has no overtravel protection: if the probe input never trips, WAITIN simply times out, but by then the move has already been commanded to its full programmed end point, and nothing retracts automatically. PROBE avoids both problems entirely. WAITIN remains right for its intended purpose — a guard switch, an air-pressure signal, an ATC's "clamped" confirmation — just not a touch-off measurement.

Results from a completed probe read back through #probe_hit and #probe_x/y/z/a (machine coordinates, captured at the trip). Store them with TOOLLEN (tool offset table, the same one G43 Hn reads) or WORKZERO (active work coordinate system) — see toolzero.mac and workzero.mac, provided by default in the macro folder, for complete working examples.

31.1 REF POINT from a Macro

REFSET stores the current machine position as the global reference point; REFCLEAR clears it; #ref_valid together with #ref_x/y/z/a read the stored values back for use elsewhere in a macro.

> **NOTE**
> There is deliberately no REFGO verb. Going back to the stored reference point needs a travel-capped safe-Z lift first (section 7.3), and only the firmware's own safe-Z path can compute that cap correctly — a macro can't replicate it exactly, and the #tz_zsafe trap in Chapter 24 explains why feeding that setting straight into MPOS doesn't work either. Use the panel's own GO REF instead (section 7.6), directly from F3 = REF POINT, or bound to any assignable key (section 4.3) for one-press access from a macro-driven workflow.

Chapter 32 — Sample Macros

Eight complete, working examples ship by default in the macro folder (0:/macros/) on the SD card. All eight are safe to read, run, and adapt as starting points for a custom routine. Seven reference I/O only through mapped function names, so they keep working after the physical wiring changes; safecheck.mac is the deliberate exception, explained below. None names a raw pin number for motion or reference logic either, so rewiring never quietly breaks one of these examples.

32.1 toolchange.mac — Manual Tool Change

The M6 handler for tool_change = MACRO. It skips the whole sequence if the requested tool is already loaded; otherwise it stops the spindle and coolant, allows a short dwell for the spindle to wind down, retracts to machine Z0 for safety, moves to a fixed tool-change position (machine X0 Y0), prompts the operator to fit the new tool, records it as current, and returns control cleanly to the calling program. Adapt the MPOS lines at the top to match the actual tool-change position on your machine.

32.2 toolzero.mac — Automatic Tool-Length Touch-Off

Measures the fitted tool's length against a tool setter and stores it in the tool table. The firmware's PROBE verb does the measurement itself — the Motion-core-timed trip, capture, and stop; this macro only decides where to probe, what to subtract, and where to file the result. It refuses to start if the probe is already showing tripped, stops the spindle and coolant first, then either travels to a stored setter position (use_pos = 1, a fixed setter shared by an ATC) or probes wherever the tool currently sits (use_pos = 0, operator jogs to the setter by hand), before prompting the operator to confirm and press CYCLE START. The resulting tool length — probe position minus the setter's known height (sensor_height) — is stored against the current tool with TOOLLEN. CONFIG → MACHINE → TOOL ZERO (section 12.5) describes the physical setter; the macro file itself shouldn't normally need editing. Like the other shipped macros, it retracts with a plain MPOS Z0 (machine top), not #tz_zsafe — see the frame trap warning in Chapter 24.

32.3 workzero.mac — Work Zero Through a Touch Plate

The touch-plate counterpart to toolzero.mac: same PROBE cycle, same CONFIG → MACHINE → TOOL ZERO settings, but the result is filed into the active work coordinate system with WORKZERO instead of the tool table, since a work zero belongs to the current setup rather than a specific tool. sensor_height is reused for a different purpose here — it holds the plate's thickness rather than a fixed setter height — so a machine is expected to use either toolzero.mac or workzero.mac, not both. After probing, the active work coordinate's Z reads exactly zero at the workpiece surface, under the plate.

32.4 refset.mac — Store the Reference Point

Stores the current machine position as the global reference point (section 7.6), with a CONFIRM prompt naming whatever reference point it's about to overwrite, so a previously stored point is never lost by accident. Going back afterward is the panel's own GO REF, not a macro verb — see the NOTE at the end of section 31.1 for why.

32.5 park.mac — End-of-Job Park

A simple end-of-job routine: stops the spindle, flood coolant, and mist together, retracts Z to the machine's top position, moves to a designated home corner, and turns off the MACRO_OUT1 output — the same "enabled" beacon safecheck.mac turns on — so an operator can tell at a glance the machine has finished and is safely parked and idle.

32.6 safecheck.mac — Guard Interlock with Retry

A retry-loop example: checks a guard/interlock input and, if open, uses CONFIRM to ask the operator to close it — retrying up to 3 times before raising an ALARM if it's still open. Once satisfied, it turns on MACRO_OUT1 as an "enabled" beacon. For this bench demo the guard is read directly as #in5, a raw pin number, rather than a mapped function name, so the macro can be tried without any I/O mapping configured first — on a real machine, map the guard switch in MAPPING and reference it by name instead. Otherwise a good reference for seeing SET, IF..GOTO, CONFIRM, and ALARM used together in one routine.

32.7 airblast.mac — Counted Output Pulses

A pure I/O example with no motion at all: pulses the MACRO_OUT2 output five times, using a counted GOTO loop with DWELL for timing between pulses. A convenient template for any chip-clearing or similar counted-pulse air-blast cycle.

32.8 atc6.mac — 6-Pocket Automatic Tool Changer

A complete working example of a linear-cassette automatic tool changer. Each pocket's X position is computed from the requested tool number (#x0 + (n-1)*#pitch), so one routine serves all six pockets: it first returns whatever tool is in the spindle to its own pocket, then collects the requested tool, driving a pneumatic drawbar on MACRO_OUT1 with carefully timed DWELLs. All moves are in machine coordinates. To adapt it to a real machine, set the geometry variables at the top (#x0, #pitch, #forky, #clearz, #forkz), then rename the file to toolchange.mac so it's picked up automatically as the M6 handler.

Chapter 33 — Macro Quick Reference

Summarizes every verb, variable, and substitution rule from Chapters 21 through 30, for quick lookup while writing or debugging a macro.

Category	Verbs
Motion (blocking)	RAPID MOVE MPOS RAPIDR MOVER
Probing (blocking)	PROBE <axis><dist> [NOERR] [AWAY]
Store a result	TOOLLEN <h> = <expr> WORKZERO <axis> = <expr> REFSET REFCLEAR
Spindle / I/O	SPINDLE CW|CCW|OFF COOLANT ON|OFF MIST ON|OFF OUT <n> ON|OFF
Timing / input	DWELL <s> WAITIN <n> HIGH|LOW [T<s>]
Operator / flow	MSG PROMPT CONFIRM ALARM SET IF..GOTO GOTO END
Labels / comments	:label ; comment
System vars	#treq #tcur #tool #x #y #z #a #in0..#in15 #in_<NAME>
Probe results (RO)	#probe_hit #probe_x #probe_y #probe_z #probe_a
Ref point (RO)	#ref_valid #ref_x #ref_y #ref_z #ref_a
Tool-zero cfg (RO)	#tz_sensor #tz_x #tz_y #tz_fixed #tz_pulloff #tz_zsafe #tz_dist
Tool offset (RO)	#tlo (offset currently in force) #tool_ofs (tool table entry for #tool)
Substitution	[expr] and #var in coordinates; %var in messages
Pin naming	IN0..IN15, OUT0..OUT15 (zero-based), or a mapped function name

NOTE

Read-only variables are refused at parse time if a SET statement targets them, so a macro can never silently fake a measurement it never took.

Appendix A — Config File Reference

Every configuration file on the SD card and the top-level section keys inside each one, as a quick cross-reference to the detailed settings tables in Chapter 12.

File	Section Keys
machine.cfg	x, y, z, axis4, general, homing, homingseq, limits, spindle, toolzero
controller.cfg	ui, display, clock, keys
stock.cfg	stock
io.cfg	io
offsets.cfg	wcs, tools

NOTE

Every configuration file carries its own version= line. A file written by a newer firmware version than the one running is refused outright rather than partially read, to avoid misinterpreting settings it doesn't understand. A file from an older version is accepted automatically, migrated to the current menu, and rewritten at the current version — with the caveats in Chapter 20 about verifying migrated homing coordinates by eye.

Appendix B — Quick Reference Card

A single-page summary of the most common operator actions and the panel key sequence for each one.

Want to…	Do This
Home the machine	Homing
Reference without switches	long Homing, then Homing
Set work zero on an axis	Axis Zero key
Type an exact work coordinate	long axis Zero key (max ±16777.216)
Change work system	WCS, then OK (long OK also goes there)
See machine coordinates	WCS-T
Load a program	File, select, OK
Preview the toolpath	VIEW (F2), long F5 to fit
Run	Start
Pause / resume	Pause, then Start
Run one line at a time	BLOCK RUN on (key, input, or MACHINE ▸ GENERAL), then Start per line
Back out of a cut	Pause, then F3 (REVERSE)
Abort	Stop
Start mid-program	Type the line number, OK, confirm, Start
Change feed	Feed-O, type, OK
Set jog speed	F4 on the main screen
Jog off a tripped limit	Limit-O, jog clear, release it
Check wiring	MENU ▸ SYSTEM ▸ DIAGNOSTICS
Run a macro	MENU ▸ MACROS, select, OK
Run a macro without the menu PIN	Set F3 key action = RUN MAC, then F3
Probe / touch off Z	MDI G38.2 Z-30, or run workzero.mac
Measure a tool length	Run toolzero.mac
Write a tool length by hand	MDI G10 L1 P<tool> Z<value>
Set the clock	MENU ▸ CONTROLLER ▸ CLOCK
Change the menu PIN	MENU ▸ CONTROLLER ▸ SECURITY, F3
Enter a licence code	MENU ▸ CONTROLLER ▸ SECURITY, F4
Find this controller’s DEVICE ID	SECURITY page, or ABOUT

Update pendant firmware	MENU ▸ SYSTEM ▸ PENDANT UPDATE, F3
See which pendant is connected	ABOUT, or the PENDANT UPDATE page

End of Document