

CNC G-code Controller

User manual — LPT port based CNC control

1. Overview

The application is a professional CNC controller program that drives stepper motors through the LPT (parallel) port. It supports both milling machine (XY plane) and lathe (ZX plane) modes. A built-in G-code editor, real-time toolpath display and workpiece profile visualization are available.

2. Main window layout

2.1 Control buttons

- **▶ Start** — Start the G-code program
- **⏸ Pause** — Pause / resume execution
- **■ Stop** — Stop the program, abort motion
- **⚙** — Open the Settings window
- **📁** — Load a G-code file (.nc, .gcode, .tap, .txt)
- **💾** — Save the current G-code to a file

2.2 DRO (Digital readout)

Shows the current position of the X, Y, Z axes in mm with 3 decimal places. Clicking a position value allows manual editing (e.g. workpiece zeroing). Each axis has a Home (⏪) and a zero (0) button. The "ALL HOME" button sends all axes to the limit switches.

2.3 Manual motion (Jog)

- **Arrows (▲▼◀▶)** — Continuous motion in the X/Y (mill) or Z/X (lathe) plane
- **Z▲ / Z▼** — Z axis motion (mill mode) or separate Z button (lathe)
- **Shift + Arrow** — Rapid (fast) motion
- **Ctrl + Arrow** — Stepping (one step increment)

2.4 G-code editor

The editor in the lower left panel lets you write and modify the G-code program directly. While running, a highlighted list replaces the editor, with the current line shown in yellow.

2.5 MDI (Manual command)

In the "MDI" input box a single G-code line can be sent for execution (Enter). Use the ↑/↓ arrows to navigate the history.

3. Toolpath view and workpiece visualization

The right-hand panel shows the 2D view of the toolpath:

- **Yellow dashed** — Rapid (G0) move
- **Blue line** — Linear (G1) feed move
- **Cyan line** — Circular (G2/G3) feed move
- **Orange line** — Approach
- **White line** — Retract
- **Green line** — Segments already executed
- **Red crosshair** — Current tool position

3.1 Workpiece profile (lathe)

In lathe mode the workpiece cross-section computed from the cutting contour is displayed with a semi-transparent fill. Mirrored around the axis line (orange dashed), both halves are visible, like the section of a real turned part.

3.2 Thread display

For G76 threading, V-grooves are drawn in the thread zone based on the pitch (P) parameter. The mirrored side shows a half-period offset — exactly like the cross-section of a real screw thread. Multiple separate threads are displayed correctly.

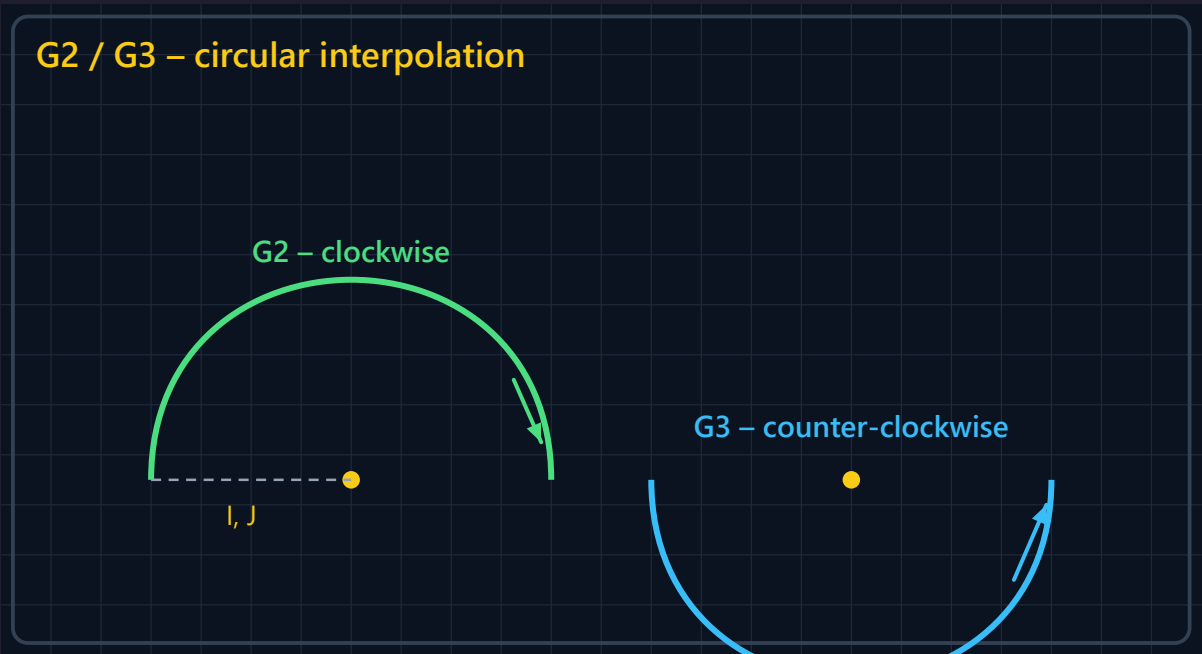
3.3 3D toolpath view (interactive)

In mill mode, the button in the top-left corner of the toolpath view switches to 3D view. The 3D view provides interactive camera control:

Mouse action	Function
Right button drag	Rotate (Yaw/Pitch) — orbit the camera around the workpiece
Middle button drag	Pan — shift the image in the screen plane
Wheel (scroll)	Zoom in / out — in 15% steps

The default view is isometric (yaw=225°, pitch=30°). The camera orbits around the center of the toolpath bounding box.

4. Supported G-code commands



4.1 Motion commands

Parancs	Description
G0 X.. Y.. Z..	Rapid positioning
G1 X.. Y.. Z.. F..	Linear interpolation at the given feed
G2/G3 .. I.. J..	CW/CCW arc with I/J center offset (XY plane)
G2/G3 .. I.. K..	CW/CCW arc with I/K center offset (XZ plane, lathe)
G2/G3 .. R..	CW/CCW körív sugár megadással (R>0: short arc, R<0: long arc)
G4 P..	Dwell — P value in seconds
G33 X.. Z.. K..	Spindle-synchronized straight motion. K = thread lead (mm/rev), Q = start angle (°). Encoder

Q.. required.

G33.1	Z..	K..	Rigid tapping right-hand (M3). K = lead, Q = peck depth (optional).
Q..			
G33.2	Z..	K..	Rigid tapping left-hand (M4). K = lead, Q = peck depth (optional).
Q..			

4.2 Mode commands

G90	Absolute positioning
G91	Incremental positioning
G61	Exact stop mode
G64 P..	Continuous path mode (P = tolerance mm)
G7 / G8	Diameter mode (G7) / Radius mode (G8) — interpretation of the lathe X axis
G53–G58	Coordinate system selection (G54 = default workpiece zero)
G94	Feed per minute mode (F = mm/min) — default
G95	Feed per revolution mode (F = mm/rev) — recommended for lathe operations
G96 S..	Constant surface speed (CSS) on — S = surface speed m/min (RPM changes automatically with the diameter)
G97 S..	Fixed RPM mode — CSS off, S = rev/min (RPM)
G50 S..	Maximum spindle speed limit (for G96 CSS mode — protects the spindle at small diameters) Handles the S parameter only, NOT usable for coordinate setting (see note below).
G48	Cancel spindle speed limit

4.2.1 Constant surface speed (G96 CSS) — details

In lathe mode the G96 command maintains a constant surface speed. The controller automatically changes the spindle RPM based on the current workpiece diameter:

$$\text{RPM} = (S \times 1000) / (\pi \times \text{diameter_mm})$$

At small diameters the RPM can get very high — therefore **always use a G50 S.. limit** before G96!

Example: G50 S3000 (max 3000 RPM)

Example: G96 S200 M3 (200 m/min surface speed, CW)

Example: G97 S1500 M3 (switch back to fixed 1500 RPM)

⚠ G50 note: On some industrial controls (e.g. older Fanuc series) G50 is also used for coordinate setting (similar to G92). **In this controller G50 is exclusively a spindle speed limit (with the S parameter).** For coordinate setting use G92 (→ chapter 25) or G10 L2 (→ chapter 20).

4.3 M-codes

M0	Program stop (mandatory) — execution halts
M1	Optional program stop — pauses, continues on Start
M3 S..	Spindle on CW (clockwise), S = speed
M4 S..	Spindle on CCW
M5	Spindle stop
T.. M6	Tool change — the T word selects the tool, M6 performs the change (macro based, see chapter 17)
M7 / M8	Coolant on (mist / flood)
M9	Coolant off
M2 / M30	Program end

4.4 Threading (G76)

The G76 cycle performs multi-pass threading with spindle-synchronized motion. The depth of cut decreases automatically (constant chip cross-section).

Parameter	Description
-----------	-------------

X	Thread root radius (final X position)
Z	Thread end (Z position)
P	Thread pitch in mm
H	First pass depth (mm, radius)
I	Infeed angle (degrees, 0 = radial)
R	Starting X radius (thread crest)
K	Starting Z position
Q	Number of spring passes
C	X safety clearance
T	Taper (X change over the Z length)

Example: G76 X17 Z-16 P1.5 H0.3 I29.5 C2 Q2

4.5 G33 — Spindle-synchronized motion

The G33 command moves the tool in sync with the spindle rotation. A spindle encoder (A/B + Index channel) is required.

G33 X.. Z.. K.. Q.. — K = displacement per revolution (mm/rev), Q = start angle (°, optional). If both X and Z are programmed, the motion runs along a straight line in the XZ plane and K applies to the axis travelling the longer distance (taper thread). For consecutive G33 lines without Q the synchronization is kept (variable-lead thread).

4.6 G33.1 / G33.2 — Rigid tapping

G33.1 Z.. K.. Q.. — Right-hand (M3) rigid tapping. **G33.2 Z.. K.. Q..** — Left-hand (M4) rigid tapping. Z = hole bottom, K = thread lead, Q = peck depth/cycle (optional, 0 = single pass).

Cycle steps: 1) Wait for Index pulse + sync. 2) Z down synchronized. 3) Spindle reversal at the bottom. 4) Z up synchronized back. 5) Spindle restored. When Q is given (chip breaking) the tapping is split into Q-deep passes: after each pass the tool fully unscrews in phase-lock back to the start point (chip evacuation), then taps to the next, deeper target — all the way to the hole bottom.

Example: G33.1 Z-20 K1.5 (single-pass M8 tapping, 20mm deep)

Example: G33.1 Z-20 K1.5 Q5 (chip breaking: full retract every 5mm, 4 passes)

5. Settings

The Settings window (⚙ button) is a scrollable page divided into sections. Changes take effect with the **Save** button; **Cancel** discards them. Next to several fields live LEDs show the current machine state. The content of each section changes depending on the selected connection type (LPT / GRBL / STB4100 / Mesa).

5.1 Machine type and tool change mode

- **Machine type** — Mill (XY plane) or Lathe (ZX plane). In lathe mode the DRO, the toolpath view and the CAM switch to ZX coordinates.
- **Diameter mode** — lathe only: the X value is interpreted and displayed as a diameter (twice the internal radius value).
- **Tool change (M6)** — *Manual*: the program stops at the WAIT point of the M6 macro and continues on Start (the machine does not return anywhere). *Automata*: the M6 macro runs without stopping.

5.2 Connection type

- **LPT (parallel port)** — the classic mode with software step generation (see 5.3).
- **GRBL (USB serial port)** — COM port selector (🔄 refresh button) and baud rate. The LPT pin, PWM and encoder settings are not used in GRBL mode (details: chapter 32).
- **STB4100 (RNR ECO MOTION 2.0)** — USB HID card on the direct Windows HID layer (without the Mach3 DLL). The panel offers live connection state, READ-ONLY input monitor (I1–I4 LEDs), HID write-cost measurement ([WRITEBENCH]), circle test (F300/F600/F900 — **moves X+Y!**) and pace-ceiling measurement ([PACECEIL], **moves X!**) (details: chapter 33).

- **Mesa 7i96S (Ethernet)** — LBP16 protocol, UDP 27181. Settings and tools:

- **IP address**, 🔍 **Find card** (automatic discovery), **Connect / Disconnect** button, live connection LED and status text.
- ⚙️ **StepGen report**, 📌 **Pinout** report (global GPIO indices), 🛡️ **Safe initialization**.
- **Jog teszt** (StepGen0–4, steps/s) and **Motion test** (signed step count, acceleration steps/s²) — for diagnostics only, moves the axis!
- **Spindle control** — Mode: Relay / PWM / StepDir; Max RPM, acceleration (RPM/s), steps/rev, PWM frequency, pulse (μs), spindle relay GPIO, coolant GPIO (–1 = disabled). In StepDir mode the Step/Dir signals of StepGen4 (5th axis connector) drive the spindle.
- **Safety inputs** — E-Stop, X/Y/Z limit GPIO index (–1 = disabled) with live state LEDs; enabling and NC/NO inversion use the switches of the "Inputs & Home" section.
- The axis Step/Dir signals are fixed on the StepGen0 (X), StepGen1 (Y), StepGen2 (Z), StepGen3 (A) channels — they cannot be selected in the Axes table.

5.3 LPT port settings

- **LPT base address (hex)** — base address of the primary port (default 0x378). For PCI/PCIe cards (Sunix / MosChip) typically the 0xD000–0xEFFF range; the program shows a dedicated hint for this.
- **2nd LPT port (hex, 0 = off)** — optional second port (e.g. 0x278). Extra outputs/inputs: pin 20–27 (Data2), pin 30–33 (Control2), input 20–27 (Status2).
- **Max step freq (Hz)** — upper limit of axis stepping (typically 60 000–250 000 Hz). If exceeded, the motion speed is limited.
- **Step (μs) / Dir (μs)** — width of the step pulse and setup time of the dir signal. The minimum period is 2×Step, so the max frequency = 1e6 / (2×Step) Hz (e.g. 10 μs → 50 kHz, 5 μs → 100 kHz). If the pulse width conflicts with the Max step freq value, a yellow warning appears.
- 🔍 **Port test** — writes to and reads back both ports; shows whether the InpOut driver and the card are reachable.
- ⚙️ **Latency benchmark** — measures LPT write latency (min / average / P99 / max / jitter). Based on this you can decide whether the card is suitable for long (10+ min) threading runs. **Recommended:** Brainboxes (UC-475, PX-157, PCI-1L). **Avoid:** onboard SuperIO LPT, Sunix / MosChip PCI(e) in compatibility mode — high jitter, risk of step loss (details: chapter 16.4).
- **CncLptBatch driver** — if the custom kernel-mode batch driver is installed, the program automatically uses it for port writes (fewer system calls, lower jitter). Its state is visible in the log.

5.4 Axes table

- **Step pin / Dir pin** — LPT output pin assignment per axis; ⌘ **Step** / ⌘ **Dir** — signal inversion.
- **Steps/mm** — resolution of motor + drive (microstep × step/rev ÷ screw lead).
- **Acceleration (mm/s²)** — per-axis acceleration. The smallest positive value gives the acceleration for manual jogging and global path planning.
- **Max mm/m** — axis max feed. The smallest positive value gives the jog rapid and the G0 speed.
- **Backlash** — backlash compensation (mm); on direction change the program adds it automatically.
- **Soft min / Soft max** and 🛡️ **Soft Limits enabled** — software limits (details: chapter 22). The Y row is hidden on a lathe.
- **A axis (4th axis)** — rotary axis around X, enabled with a checkbox. Parameters: steps/degree (e.g. 16000 step/rev ÷ 360 = 44.444), acceleration (°/s²), max feed (°/min), backlash (°). On LPT the Step/Dir may also go to the Control port of port 1 (C0–C3 / pin 10–13).

5.5 Manual motion, path smoothing, acceleration profile

- **Step increment Ctrl + arrow (mm)** — incremental jog step.
- **Jog slow feed (mm/min)** — speed of continuous manual motion; on save it is clamped to the jog rapid.
- **Path smoothing (G64)** — **Tolerance (mm)** — chord-deviation tolerance. Larger value → faster passage through corners → higher axis acceleration → risk of step loss. Typically 0.005–0.05 mm. If the value is too large relative to the accelerations, a yellow warning appears.
- **Look-ahead** — number of segments planned ahead for merging the velocity profile.
- **Ramp type** — *Trapezoid*: classic linear acceleration. *S-curve (smooth)*: jerk-free start/stop, less resonance and step-loss risk, slightly longer ramp time for the same ramp length.

5.6 Spindle / Coolant / Encoder

- **Spindle pin / Coolant pin** + inversion — M3/M4/M5 and M7/M8/M9 outputs (LPT/STB4100; for Mesa on the Mesa panel).
- **Spindle drive type** — three modes:
 - **On/Off** — simple relay on/off, no speed control.
 - **PWM** — software PWM signal for a VFD or ESC. PWM pin, inversion, frequency (Hz), min/max RPM.
 - **Step/Dir servo** — e.g. Beckhoff AX2010/AX2526. Step/Dir/Enable pins and their inversions, steps/rev (100–200 000), acceleration (RPM/s), max RPM. Supports: M3 (CW), M4 (CCW), M5 (ramped stop), S while running, spindle override slider.
- **Spin-up/down time (s)** — after spindle ON no motion starts for this many seconds; after M3/M4 the next G-code line also runs only after this wait (default 3 s, applies to every drive type).
- **Index homing (M5/M30 → C axis 0)** — Fast search (RPM, CW, typically 60–300) and Slow approach (RPM, CCW, typically 5–30) to the exact edge of the index pulse (details: chapter 34).
- **Tapping (G33.1 / G33.2 / G84) protection** — *Reverse lead factor*: the reverse command is sent ahead by this multiple of the coast-down (0.5 = halfway through coast, 1.0 = start of coast). *Overshoot tolerance (threads)*: Z follows the physical spindle position at the hole bottom up to this many threads deep, beyond that E-Stop (typically 1.0–2.0). *Watchdog (revolutions)*: after reverse, the direction change must occur within this many revolutions, otherwise E-Stop (typically 1.0).
- **Encoder** — Index signal pin, pulses/rev, index inversion; optional A/B quadrature channels (pin, CPR, A/B inversion; 4× resolution, e.g. 100 CPR = 400 edges/rev). Required for threading and G33 sync. In Mesa mode the dedicated Encoder channel of the 7i96S can be used (enable switch, live A/B/Index LEDs).
- **Mach3-style "blind" threading** — enabled: at the start of every pass RPM measurement + wait for index pulse, then it runs through at a fixed computed step rate (more stable at high RPM, but does not follow spindle slow-down). Disabled (default): continuous encoder sync along the whole pass.

5.7 Inputs & Home

- **Home switches** — X/Y/Z home pin; **Limit switches** — X/Y/Z limit pin + "Limits active" switch; **E-Stop** pin + "E-Stop active".
- **Inv. Home / Inv. Limit / Inv. E-Stop (NC)** — for normally-closed (NC) contact switches.
- **Home direction** — ✓ = searches in the positive direction.
- **Homing speeds** — Fast (mm/min), Slow (mm/min) for the exact edge, Backoff (mm) retract.
- **Home search mode** — *Unlimited*: a single unlimited search (9999 mm). *Limited*: "Attempts" number of tries, each up to the "Max search (mm)" distance; if no switch is found, the axis is skipped.
- **G28 / G30 reference positions (mm)** — X/Y/Z machine coordinates for the return-home commands (details: chapter 19).
- **Live Status port** — raw value and LEDs of the LPT1 (S3 pin15, S4 pin13, S5 pin12, S6 pin10, S7 pin11*) and LPT2 inputs in real time; for STB4100 the IO input LEDs, for Mesa the GPIO states.

5.8 Macros, Post-Processor, settings portability, log

-  **Macro editor** — editing M-code macros (M6 tool change, custom macros) (details: chapter 17).
-  **Post-Processor editor** — editing the header/footer templates of the CAM-generated G-code with a placeholder list. Takes effect when existing .cncproj projects are re-exported.
-  **Settings portability** — *Export*: the entire configuration (pin settings, tool library, quick buttons, macros, post-processor) into a single portable .cncset.zip file. *Import*: restore on another machine.
-  **Log** — the application's event log (the file: **cnc_log.txt** in the program folder). Real-time diagnostics also go here: [SegDiag] segment jitter / late steps, [MasterHotStall], [WRITEBENCH], [PACECEIL], [GC] and the thread priority / CPU core assignment lines.

5.9 Real-time operation — system-level behaviour

At startup the program **requests administrator rights** (requireAdministrator manifest). This is needed so the step generator threads can get real-time (REALTIME range) priority while the process as a whole stays in the HIGH class, so the user interface does not freeze.

- **Master step thread** — on a dedicated CPU core (last core), priority 24; this emits the pulses to the LPT port.
- **Producer (StepGenerator) thread** — on a 4+ core machine on its own dedicated core (second to last), priority 22; this plans the steps ahead.
- **Garbage collector (GC)** — during Run the GC is confined to cores 0–1 and runs with an enlarged Gen0 budget, so it does not interrupt step generation while running. For this the application restarts itself once at startup with the required environment variables (DOTNET_GCgen0size) — this is normal.
- The effect can be checked in the log: "[MasterStepGen] ... master thread prio=24 realtime=True admin=True", "[StepGen] Fix #18 producer affinity=... realtime=True", and the Late / LagStretch values in the [SegDiag] lines.

Mill settings:

The screenshot displays the CNCUI software interface for a mill. The main window is titled "CNCUI" and contains several functional areas:

- E-Stop Section:** Includes a "Start" button, a "Szű" (Filter) button, and a "Stoj" (Stop) button. Below these are buttons for "ALL HOME" and coordinate readouts for X (47,180), Y (13,750), and Z (-5,500).
- Feed Rate Controls:** Sliders for F (100%), R (100%), and S (100%) are shown, along with a feed rate of 18000 mm/min.
- Spindle Speed Controls:** A section for "Orsó / Hűtés / Jeladó" (Spindle / Cooling / Indicator) with settings for Orsó pin (D6), Hűtés pin (D7), and various invert and PWM settings.
- Axis Configuration Table:** A table titled "Tengelyek" (Axes) showing step pins, dir pins, step dir, steps/mm, velocity, max mm/min, and backlash for X, Y, and Z axes.
- G-code Section:** A text area for entering G-code, with a "G-code" button and a "Leállítva" (Stopped) status indicator.
- Bottom Bar:** A series of buttons labeled "Orsó BE", "Szerszámcseré", "Q3", "Q4", "Q5", "Q6", "Q7", "Q8", "Q9", and "Q10".

E-Stop

▶ Start

⏮ Szű

⏹ Stoj

⏏

⏪

⏩

⏴

⏵

⏶

⏷

ALL HOME

X

47,180

⬆

0

Y

13,750

⬆

0

Z

-5,500

⬆

0

G53

G54

G55

G56

G57

G58

F 100%

R 100%

S 100%

100

100

100

F

F 0 mm/min

Orsó kl

S 18000

→ 0 RPM

0%

Szeg:

6/429

(1%)

Mód: Hardver

⬆

⬅

➡

⬆

XY

➡

Z

⬆

Z

⬆

Nyitak: folyamatosan | Drift: rapid | Csat: lapos

Lépésköz: 1 mm

Váratott: 33:42

Eltelt: 00:14

Hátraérvő: --1--1--1--

G-code

1 M1 G17 G40 G90 G80

2 M1 G21

3 M1 G54

4 M1 T2 M5

5 M1 S18000.000 M3

6 (Zsebr - Zsebr)

7 M1 G0 X-13.500 Y12.250 Z5.000

8 M1 G1 X-13.500 Y12.250 Z-0.500

9 F150.000

10 M1 G1 X-13.500 Y12.250 Z-0.500

11 F400.000

M01>

Leállítva

Pálya nézet

Maró CAM

Pálya nézet (XY sík)

Beállítások

Gép típus

Maró (XY sík)

Eszterga (ZX sík)

Átmérő mód (X érték = átmérő)

Csatlakozás típusa

LPT (párhuzamos port)

GRBL (USB soros port)

GRBL soros port beállítások

COM port: COM6

Baud: 115200

A GRBL vezérlő USB-n csatlakozik. Az LPT beállítások (pin konfiguráció, PWM jeladó) nem használhatók GRBL módban.

Tengelyek

	Step pin	Dir pin	Step In Dir	Lépés/mm	Gyorsulás	Max mm/m	Backlash
X	D0 (Pin 2)	D1 (Pin 3)	☒	200	100	3000	0
Y	D2 (Pin 4)	D3 (Pin 5)	☒	200	100	3000	0
Z	D4 (Pin 6)	D5 (Pin 7)	☒	200	100	3000	0

Orsó / Hűtés / Jeladó

Orsó pin: D6 (Pin 8)

Hűtés pin: D7 (Pin 9)

☐ Orsó invertálás

☐ Hűtés invertálás

Orsó meghajtás típusa

PWM (frekvenciaváltó, ESC)

PWM pin: D2 (Pin 4)

☐ PWM invertálás

Frekvencia (Hz): 1000

Min RPM: 0

Max RPM: 24000

Jeladó (encoder) — menetszámszám / G33 szinkron mozgáshoz

Index jel: C3 (Pin 17, Selectin)

Impulzus/ford: 1

☐ Index jel invertálás

A/B kvadrátúra csatlakozás (opcionális — nagyobb felbontás)

A csatona: C2 (Pin 16, Init)

B csatona: C1 (Pin 14, AutoLF)

CPR (vonalford): 100

☐ A inv.

☐ B inv.

Makrók (M-kód)

M-kód makrók szerkesztése (M5 szerkesztőre: egyéni makrók)

Makró szerkesztő megnyitása

Orsó BE

Szerszámcseré

Q3

Q4

Q5

Q6

Q7

Q8

Q9

Q10

Lathe settings:

E-Stop

▶ Start

⏮ Szű

⏹ Stoj

⏏

⏪

⏩

⏴

⏵

⏶

⏷

ALL HOME

X

10,000

⬆

0

Z

-5,170

⬆

0

G53

G54

G55

G56

G57

G58

F 100%

R 100%

S 100%

100

100

100

F

F 0 mm/min

Orsó kl

S 500

→ 0 RPM

0%

Szeg:

6/83

(7%)

Mód: Hardver

⬆

⬅

➡

⬆

ZX

➡

Z

⬆

Z

⬆

Nyitak: folyamatosan | Drift: rapid | Csat: lapos

Lépésköz: 1 mm

Váratott: 01:52

Eltelt: 00:13

Hátraérvő: --1--1--1--

G-code

1 M1 G18 G40 G90 G80

2 M1 G21

3 M1 G8

4 M1 G95

5 M1 G54

6 M1 T04

7 M1 G97 S500 M3

8 (Menetszámszám - Menetszámszám)

9 M1 G0 X12.000 Z2.000

10 M1 G0 X10.000 Z0.000

11 M1 G76 X9.000 Z-40.000 P1.500 H0.300

12 C2.000 T15.000 Q2 129.500

13 M1 G83 U0 W0

14 M1 M5

15 M1 M9

M01>

Leállítva

Pálya nézet

Eszterga CAM

Pálya nézet (ZX sík)

Beállítások

Gép típus

Maró (XY sík)

Eszterga (ZX sík)

Átmérő mód (X érték = átmérő)

Csatlakozás típusa

LPT (párhuzamos port)

GRBL (USB soros port)

LPT alap cím (hex): 378

Z. LPT port (hex, 0=k): 278

Tengelyek

	Step pin	Dir pin	Step In Dir	Lépés/mm	Gyorsulás	Max mm/m	Backlash
X	D0 (Pin 2)	D1 (Pin 3)	☒	200	100	3000	0
Y	D2 (Pin 4)	D3 (Pin 5)	☒	200	100	3000	0
Z	D4 (Pin 6)	D5 (Pin 7)	☒	200	100	3000	0

Orsó / Hűtés / Jeladó

Orsó pin: D6 (Pin 8)

Hűtés pin: D7 (Pin 9)

☐ Orsó invertálás

☐ Hűtés invertálás

Orsó meghajtás típusa

Step/Dir szervó (Beckhoff stb.)

On/Off (relé, egyszerű)

PWM (frekvenciaváltó, ESC)

Step/Dir szervó (Beckhoff stb.)

Dir pin: 2.D1 (Pin 3)

☐ Step inv.

☐ Dir inv.

☐ Enable inv.

Step/fordulat: 10000

Gyorsulás (RPM/s): 500

Max RPM: 24000

Jeladó (encoder) — menetszámszám / G33 szinkron mozgáshoz

Index jel: C3 (Pin 17, Selectin)

Impulzus/ford: 1

☐ Index jel invertálás

A/B kvadrátúra csatlakozás (opcionális — nagyobb felbontás)

A csatona: C2 (Pin 16, Init)

B csatona: C1 (Pin 14, AutoLF)

CPR (vonalford): 100

☐ A inv.

☐ B inv.

Makrók (M-kód)

M-kód makrók szerkesztése (M5 szerkesztőre: egyéni makrók)

T4 M0.10

Orsó BE

Szerszámcseré

Q3

Q4

Q5

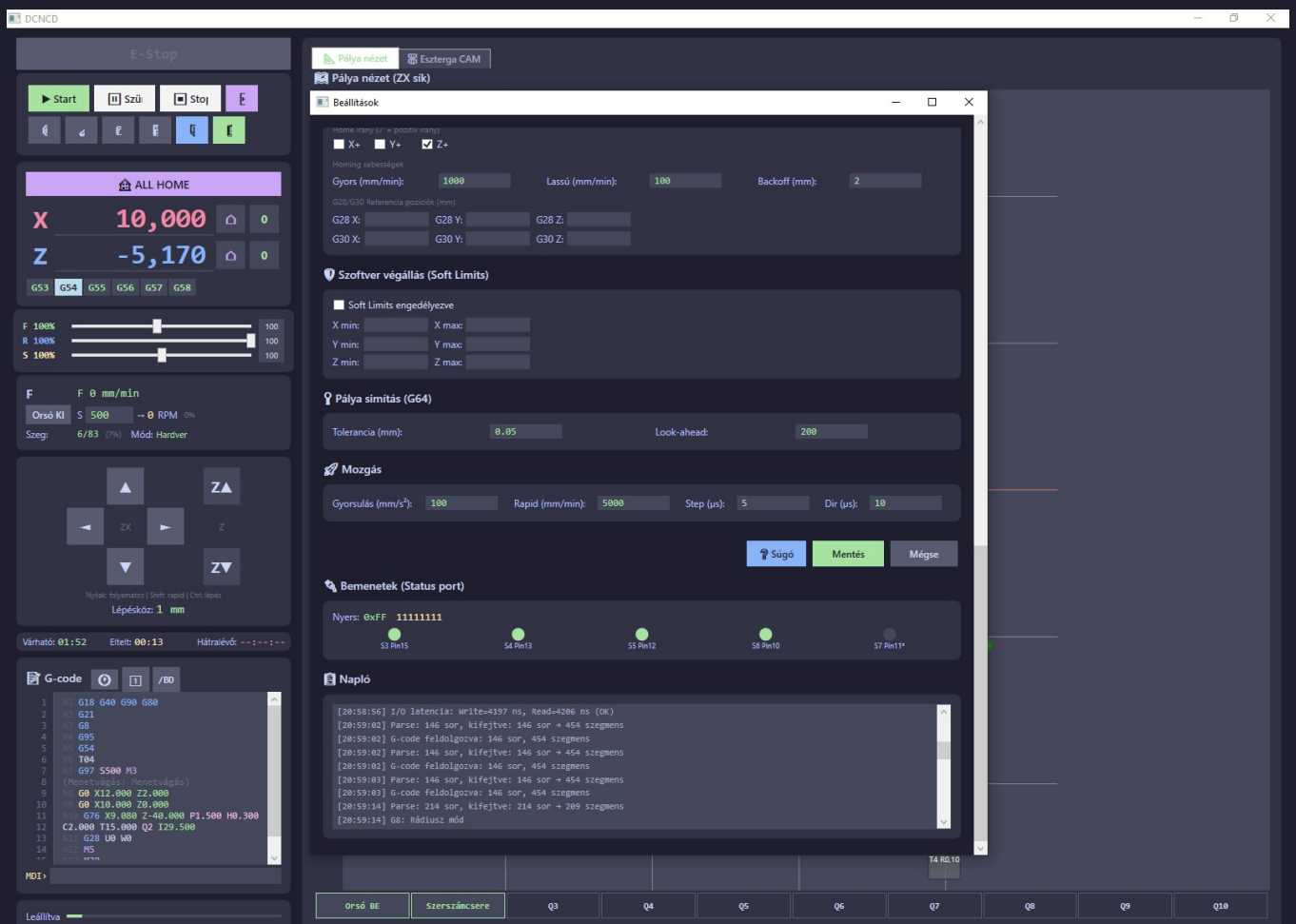
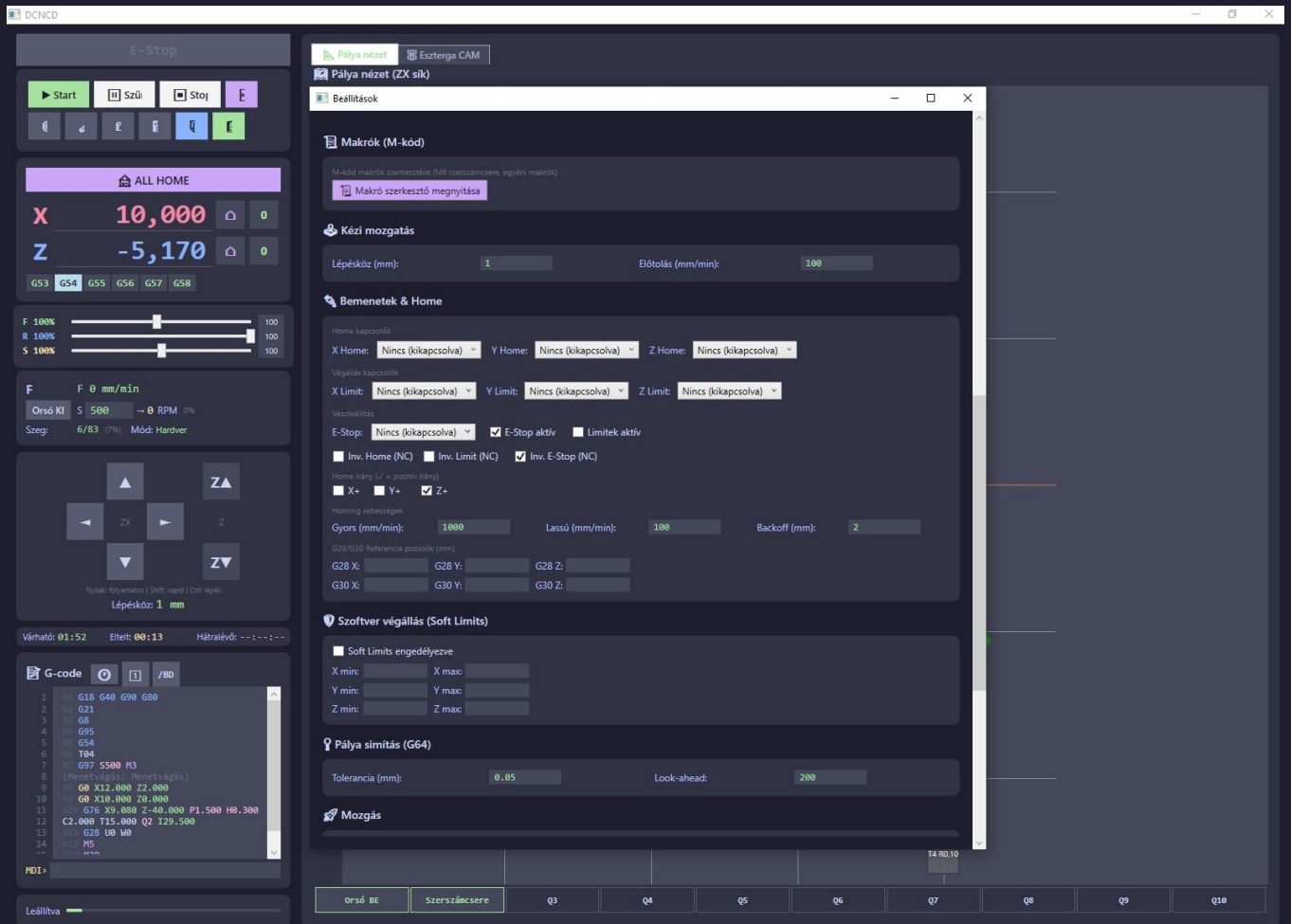
Q6

Q7

Q8

Q9

Q10



6. Keyboard shortcuts

F1	Open help (available from anywhere)
Alt + R	Start / resume program (from any tab)
Space	Pause / Resume — stops immediately, even mid-segment (from any tab)
Escape	Stop program (from any tab)
Arrow keys	Continuous manual motion (jog)
Shift + Arrow	Rapid manual motion
Ctrl + Arrow	Stepping manual motion (one step increment)
PgUp / PgDn	Z axis motion (in mill mode)
Ctrl + O	Open G-code file
Ctrl + S	Save G-code file
Alt + Q	Switch between Toolpath view ↔ CAM tab
Alt + W	Rewind to the beginning
Alt + S	Spindle on/off
Enter (in the MDI box)	Execute MDI command
↑ / ↓ (in the MDI box)	Browse MDI history
Ctrl + / -	Feed override ±10%
Shift + / -	Rapid override ±10%
AltGr + / -	Spindle override ±5%

7. Project save / load

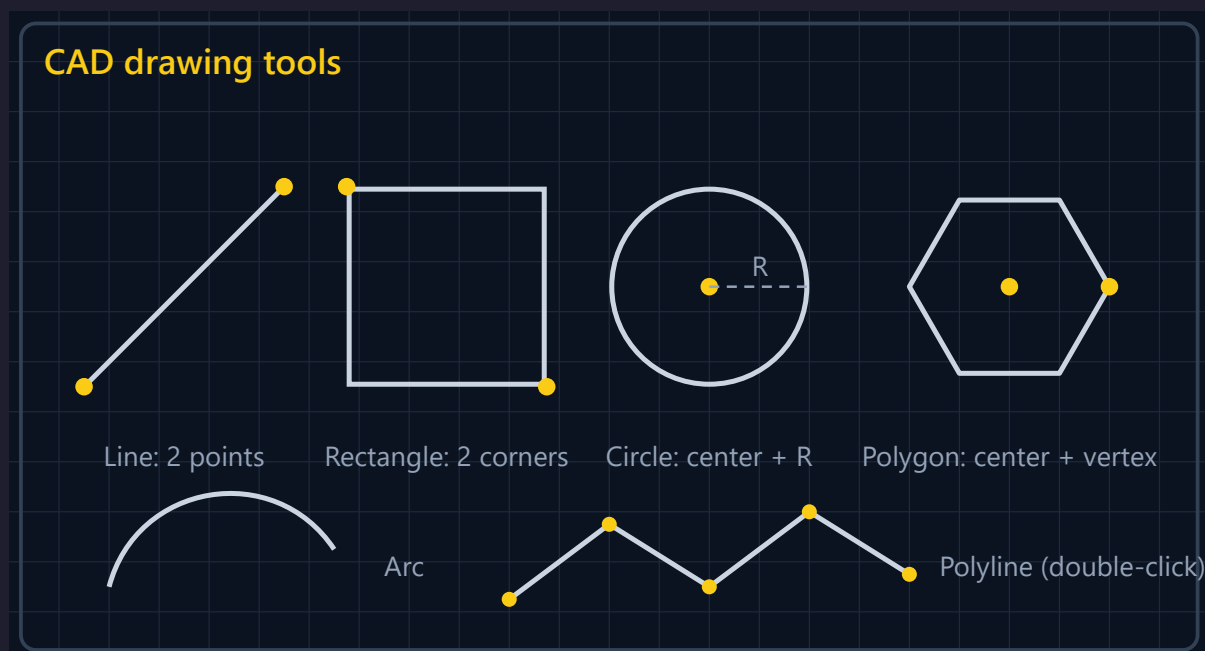
The application uses its own project format (.cncproj), which is JSON based and contains the CAD drawing, the G-code program, the machine mode and the diameter mode setting.

-  **Save project** — Saves the current state (CAD entities + G-code + mode settings)
-  **Load project** — Restores the full project state from the .cncproj file
- **Recent files** — List of recently opened G-code files for quick access

8. CAD drawing system

The built-in 2D CAD system lets you draw machining contours directly in the application. The drawn geometry can be used for CAM operations: after selection, the G-code generator automatically recognizes the contours.

8.1 Drawing tools



Tool	Description
------	-------------

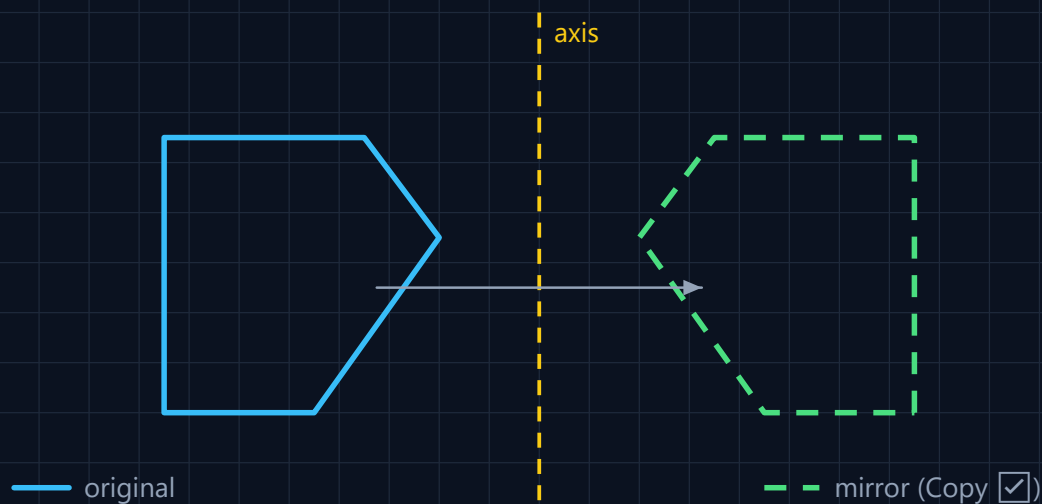
 Select	Select entities by clicking or by dragging a rectangle (Esc also switches to it)
• Pont	Place a point entity (e.g. drilling center)
/ Vonal	Draw a straight line between two points
 Rectangle	Draw a rectangle by giving two corner points
○ Circle	Draw a circle by giving center + radius
⌒ Arc	Draw an arc
● Ellipszis	Draw an ellipse
⬡ Polygon	Draw a regular polygon: 1st click = center, 2nd click = radius and angle. Default: 6 sides Editable by double-click (side count, radius, position, rotation).
▬ Polyline	Draw a polyline (multiple points, double-click: finish)
A Text	Place a text entity (for engraving)

8.2 Editing commands

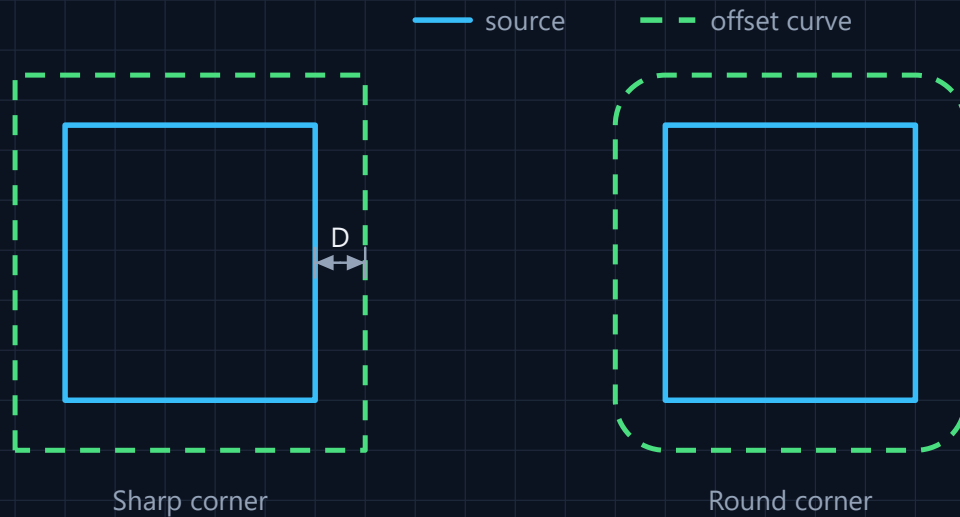
-  **Delete (Del)** — Delete selected entities
-  **Copy** — Offset copy of the selected entities
-  **Undo (Ctrl+Z)** — Undo the last operation (max 50 steps)
-  **Redo (Ctrl+Y)** — Repeat the undone operation
- **Ctrl+A** — Select all entities
- **Esc** — Abort the current drawing / clear selection

8.3 Transformations

Mirror about an axis



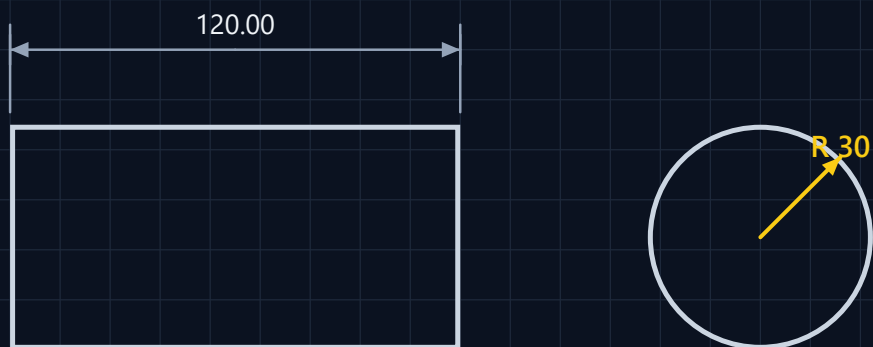
Offset – corner handling



Array – rectangular / polar / along curve



Dimensions and measuring tape



circle/arc: R / Ø; two points: distance, ΔX , ΔY

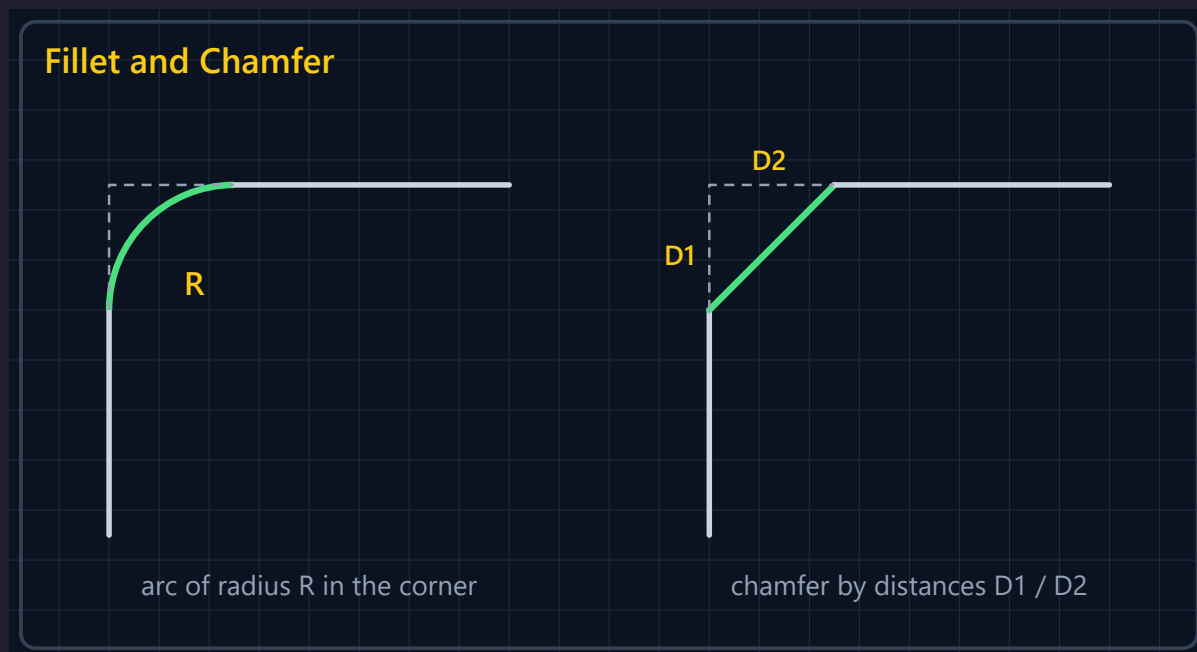
- ⇄ **Mirror X** — Mirror selected entities on the vertical axis
- ⤴ **Mirror Y** — Mirror selected entities on the horizontal axis
- ⤵ **Mirror along axis** — Mirror on an arbitrary axis. The axis can be given three ways: (1) click two points of the axis; (2) click an existing, *unselected* line — it becomes the axis; (3) type the point coordinates as **x,y + Enter** (twice). The status bar shows which point is expected. Esc: cancel.
- ☒ **Copy** (checkbox next to the mirror buttons) — When on, the originals are kept and the mirror image is created as new entities (mirror with copy); when off, the selected entities are *moved* to their mirror image. Applies to all three mirror modes.
- 🔄 **Rotate 90°** — Rotate selected entities by 90°

Typing commands (command finder)

On the canvas — when no drawing is in progress — just start typing the name or abbreviation of a button: the status bar shows the best 5 matches as **Command: ci_ → circle, ...** Diacritics are ignored ("kor" = "kör") and both Hungarian and English names work. When only one match remains the command **runs immediately**; with several matches **Enter** runs the first one, **Backspace** deletes, **Esc** cancels.

Command	Typable names / abbreviations
Drawing	select/sel · point · line/l · polyline/pl · rectangle/rect · circle/c · arc/a · ellipse/el · polygon/pol · curve/spline · text/t
Modify	trim/tr · split · measure/dist · join/j · explode/x · fillet/f
Transform	mirror x/mx · mirror y/my · mirror/mi · rotate/ro · array/ar · copy/co · move/m
Editing	delete/del/erase/e · undo/u · redo
Constraints	horizontal/h · vertical/v · coincident · parallel/par · perpendicular/perp · length/len · radius/r · equal/eq · delete constraint

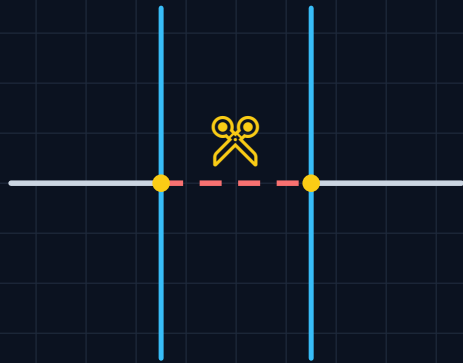
8.4 Join and fillet



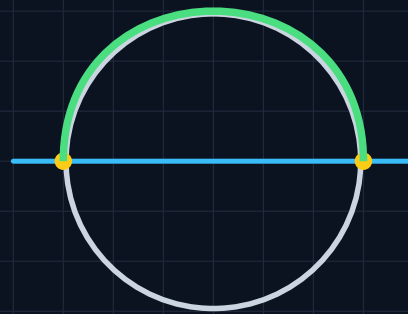
- ✂ **Join** — Join multiple lines/arcs into a single polyline
- ⤶ **Fillet** — Round the corner between two selected lines with the given radius (R: field)

8.5 Explode, Trim, Split

Trim and Split



clicked segment removed



circle → two arcs at intersections

Extend to boundary



line grows to nearest intersection

These tools let you break composite entities apart and cut geometry precisely at intersections.

-  **Explode** — Break composite entities into primitives:

- Rectangle → 4 lines
- Ellipse → 32 line segments
- Polyline → individual lines (for closed: + closing line)

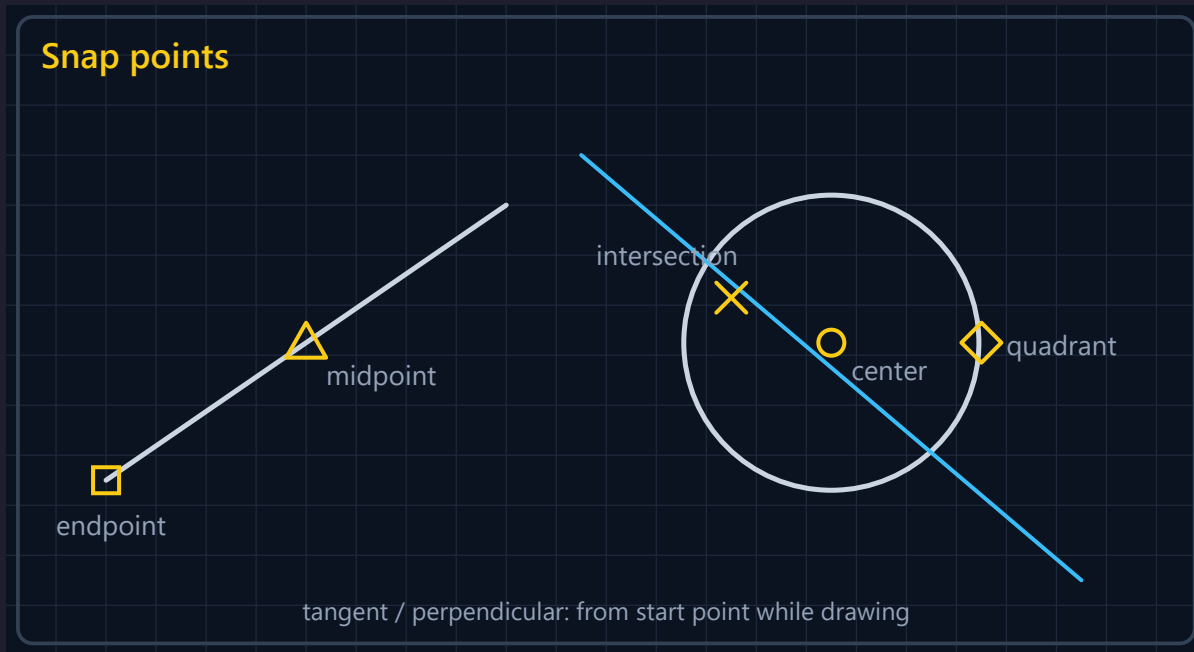
Note: a circle cannot be exploded because it can be trimmed and split directly.

-  **Trim** — Click a segment of an entity and it is cut away up to the nearest intersections. The rest remains as separate entities. If there is no intersection, the whole entity is deleted.
Supported: line, arc, circle. Rectangle/ellipse/polyline act as cutting elements (provide intersections) but cannot be trimmed themselves — explode them first!
-  **Split** — Click an entity and it is cut in two separate parts at the nearest intersection.
Circle → two semicircles (arcs), line → two shorter lines, arc → two smaller arcs. Both operations can be undone with Ctrl+Z.

Typical workflow: draw a rectangle + a circle inside it → Explode the rectangle → Trim the unneeded segments at the intersections.

8.6 Snap

Snap points



- **Grid** — While drawing, the mouse snaps to the grid
- **Endpoint** — Snap to endpoints of existing entities (line ends, circle points, etc.)

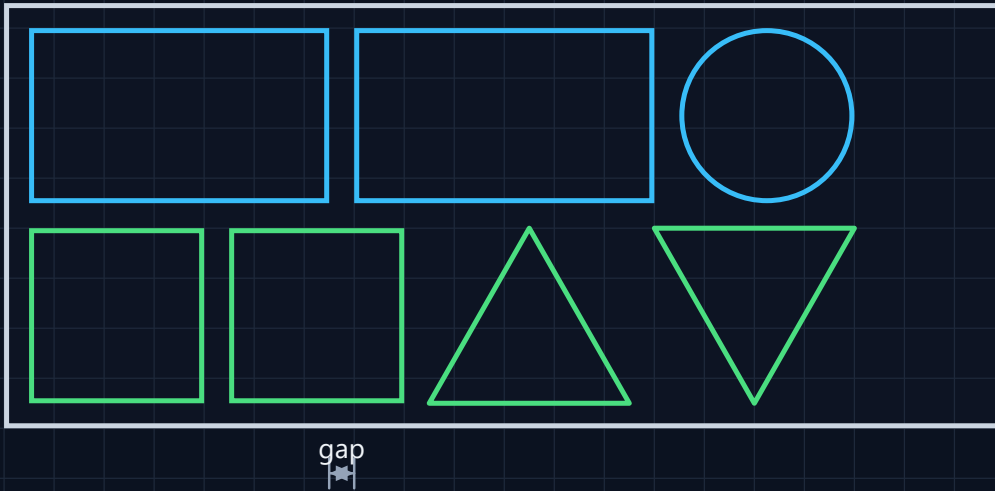
8.7 DXF import / export

The  button imports a DXF file, loading lines, circles, arcs and polylines. The  button exports the current drawing to DXF format for compatibility with other CAD programs.

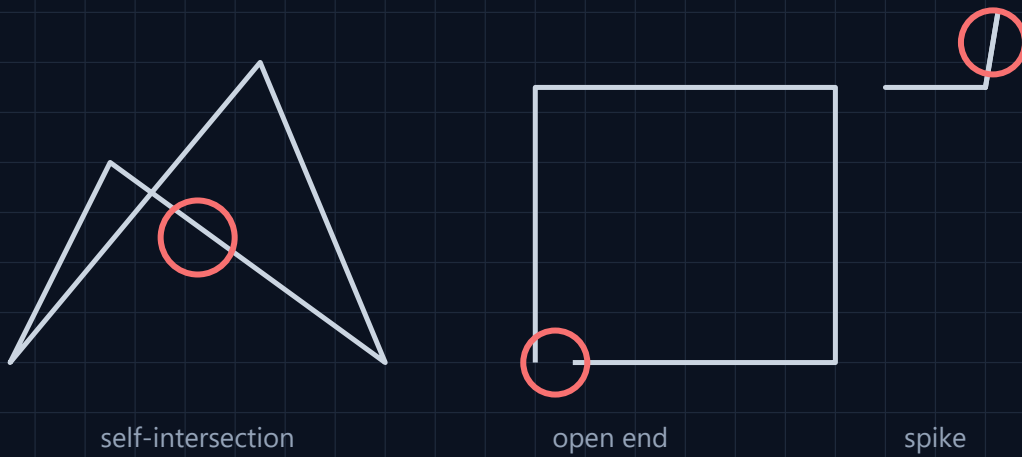
8.8 Additional toolbar functions

- **Chamfer** — Corner chamfer by clicking two segments (D1/D2 distances), or group chamfer on every corner of the selected shapes.
- **Extend** — The line is extended to the nearest boundary.
- **Offset settings** — Distance D, both sides, through point, corner mode (sharp / round / chamfer / smooth), end caps; closed-shape offset (Clipper2), delete source / source layer.
- **Mirror** — X / Y axis or an arbitrary two-point axis; with Copy ☒ the original is kept.
- **Array** — Rectangular grid (dX, dY, count) or polar array around a center.
- **Move / Copy offset / Scale** — Relative or absolute move, offset copy, proportional scaling.
- **Close (CloseCrv) and Seam (CrvSeam)** — Join and close the selected pieces; relocate the start point (CAM entry) of a closed curve.
- **Selection filters** — Select open / closed / duplicate entities, lock selected.
- **Vector Doctor** — Find broken curves: self-intersection, crossing, overlap, open ends, spikes.
- **Nesting** — Automatic arrangement of selected (or all) parts on sheet(s) with adjustable gap.
- **Dimension (Dim) and measuring tape** — Dimension line with arrows and value; measure free / horizontal / vertical, R/Ø for circles/arcs.
- **Z level (2.5D)** — Per-entity Z level / depth.
- **Import** — DXF, SVG import, project import (append to the open one).
- **Zoom** — Zoom extents / selection; Rebuild.

Nesting – parts arranged on a sheet

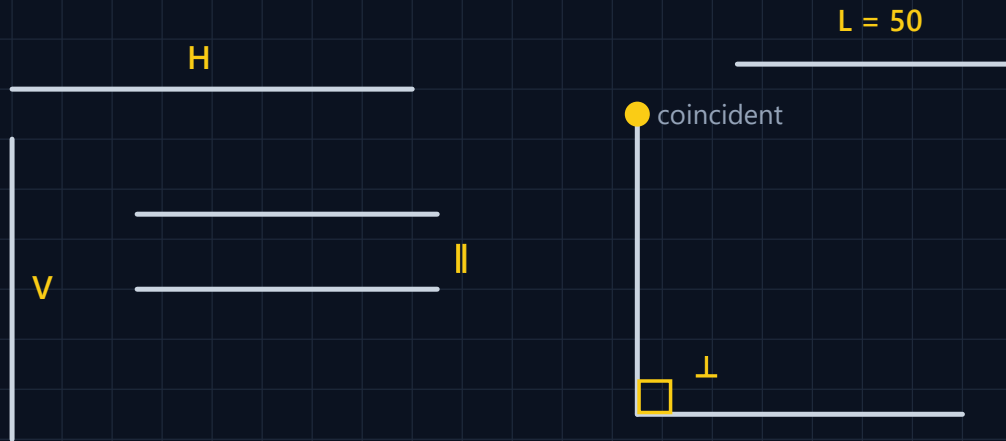


Vector Doctor – find broken curves



9. Constraint system

Constraints



The solver re-evaluates constraints after every edit

The CAD drawing supports parametric constraints for defining exact relations between geometric elements. Constraints are applied automatically while editing.

Constraint	Description
H Horizontal	Force the selected line to be horizontal
V Vertical	Force the selected line to be vertical
○ Coincident	Join two points or endpoints onto each other (coincident)
Parallel	Force two lines to be parallel
⊥ Perpendicular	Force two lines to be perpendicular
L Fixed length	Fix the length of a line to a given value
R Fixed radius	Fix the radius of a circle/arc to a given value
= Equal	Make the length or radius of two elements equal
× Delete	Remove the constraints of the selected element

10. Layer manager

Layers – separate CAM processing



Color, visibility, lock per layer

The layer panel on the left lets you organize the drawing elements into layers. Each layer has a color, visibility and lock

state.

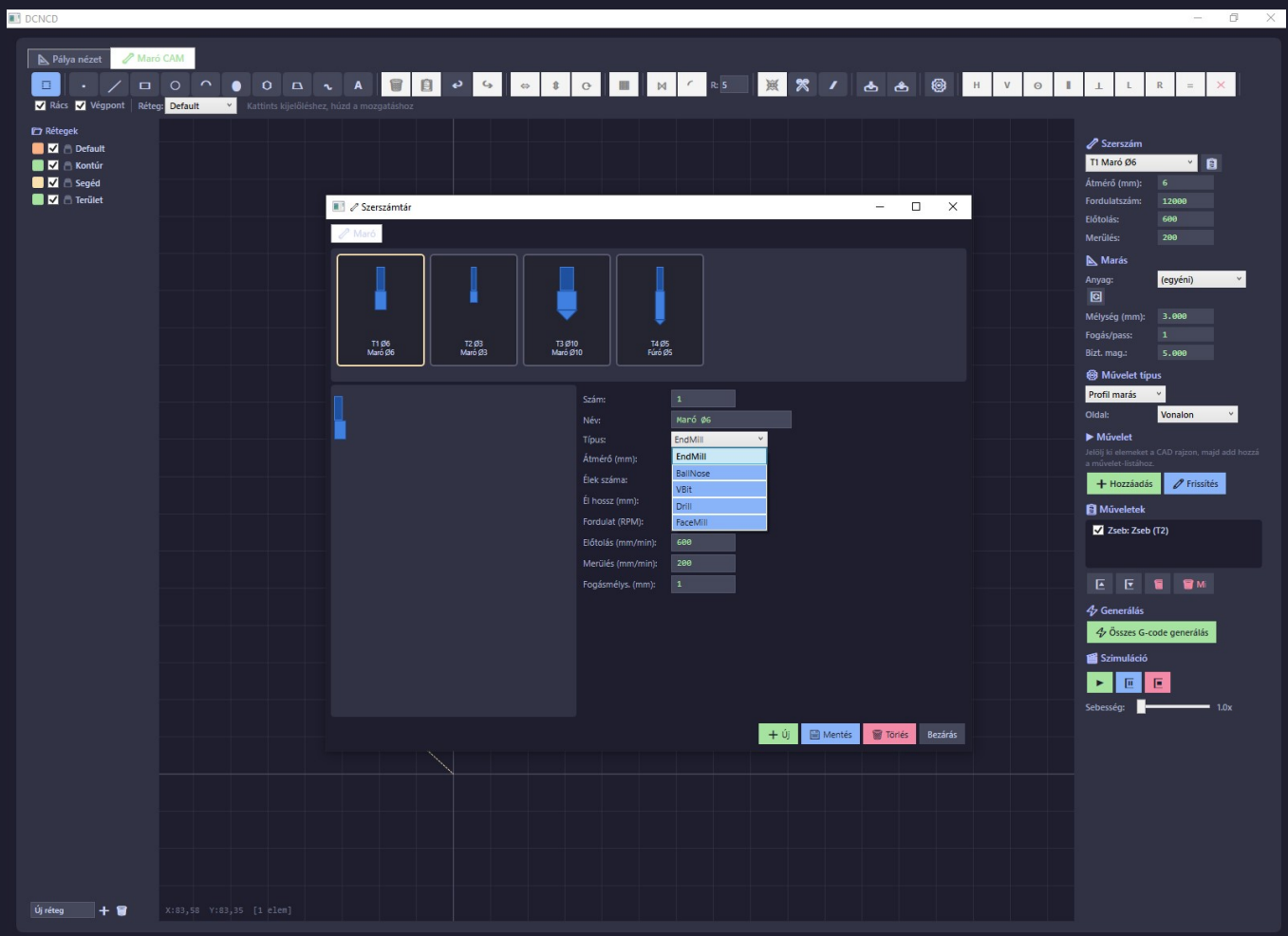
- **Color square** — Click to change the layer color
- **Eye icon (✓)** — Toggle layer visibility
- **Lakat ikon** — Lock the layer (locked layers cannot be edited)
- **+ New layer** — Add a new layer with a custom name
- **🗑 Delete** — Delete the selected layer (the Default layer cannot be deleted)
- **Active layer** — The active layer that receives new elements can be chosen on the toolbar

11. Tool library

The application has a built-in tool library for managing mill and lathe tools. The library is saved to a JSON file (tool_library.json) in the application folder.

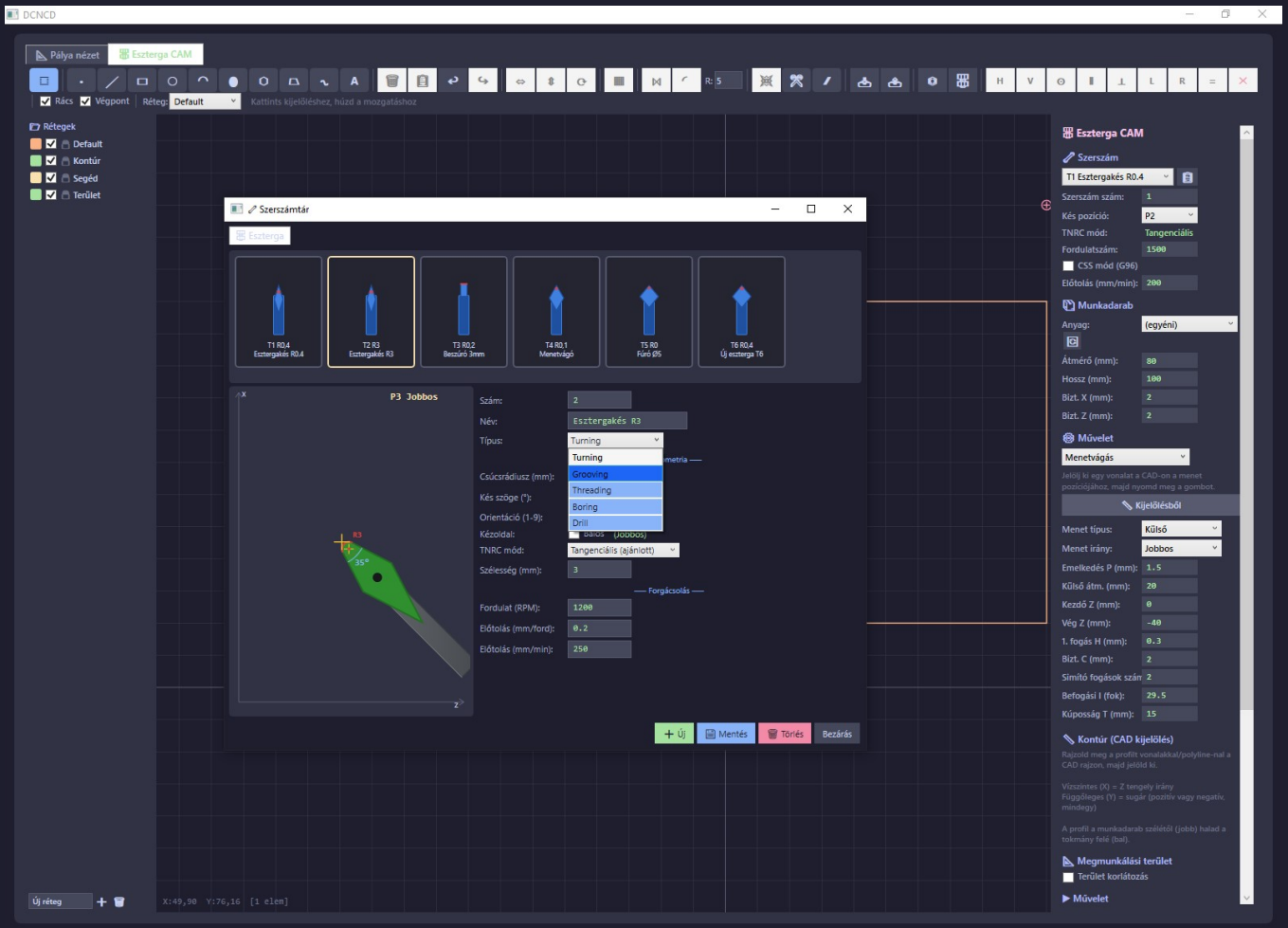
11.1 Mill tools

- Tool number, name, type (End mill, Ball, V-bit, Drill, Face mill)
- Diameter, flute count, flute length
- Spindle speed, feed, plunge rate, depth of cut, stepover



11.2 Lathe tools

- Tool number, name, type (Turning tool, Grooving, Threading, Internal, Drill)
- Nose length, width (for grooving)
- Spindle speed, feed (mm/rev and mm/min)



12. Material database

The material selector drop-down of the CAM panels offers predefined material types. When a material is selected, the application automatically computes the recommended cutting parameters (spindle speed, feed, plunge rate, depth of cut) from the tool diameter and the material properties.

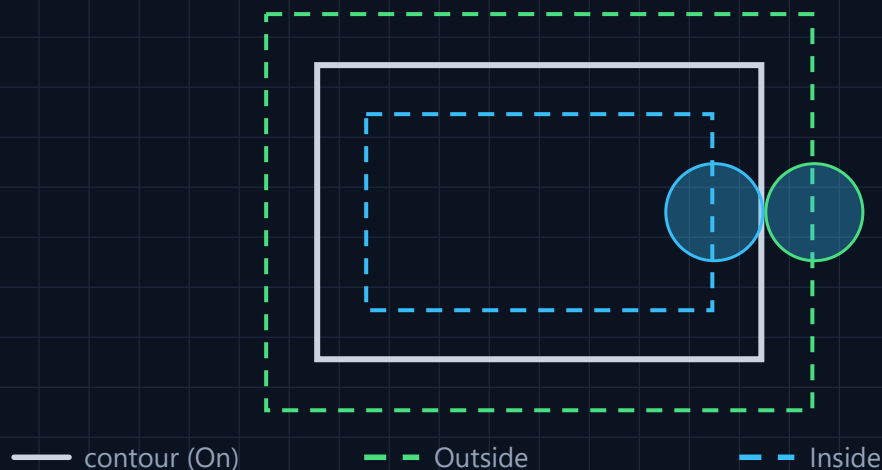
- **Mill** — Surface speed → RPM, chip load → feed, depth as a ratio of the tool diameter
- **Eszterga** — Surface speed → RPM (for the workpiece diameter), feed mm/rev, roughing depth of cut
-  **Recalculate** — Parameters can be recomputed any time from the modified tool/material combination

13. Mill CAM

On the "Mill CAM" tab you can draw the machining geometry and generate toolpaths. Several operations can be chained into a single G-code program with the operation list.

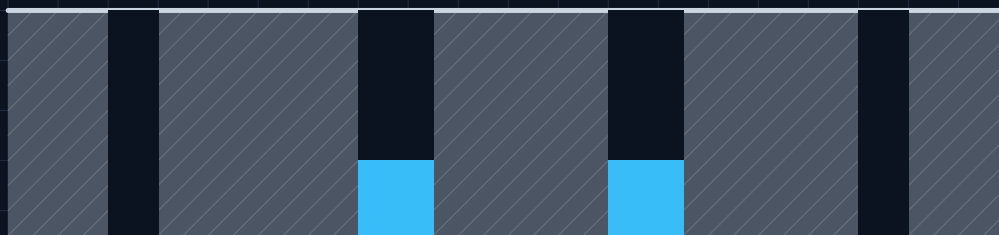
13.1 Operation types

Profile – side selection



Holding tabs

Part does not come loose on last pass



tab (TabHeight × TabLength)

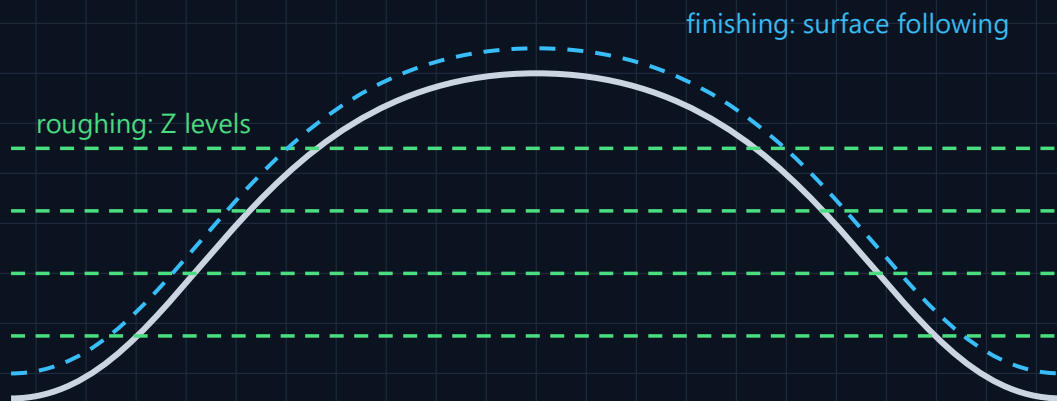
Operation	Description
Profile milling	Milling along a contour, selectable side (on line, outside, inside, left, right)
Pocket milling	Remove the material inside a closed contour (parallel/concentric strategy, overlap %, angle finishing contour). Island support: if you select several closed contours, the largest is the pocket boundary and the ones inside are handled automatically as islands — the tool avoids them.
Drilling	Drilling at point or circle centers, with adjustable peck depth (peck drilling)
Engraving	Engrave text or contour to a given depth (stroke-font based)
Slot milling	Cut a slot along a contour, widening to both/left/right side
V-Carve	Milling between two contours with a V-bit. The depth is computed automatically from the width and the V angle. The tool lifts at corners. Optional multi-step roughing.

13.1.1 3D milling

- **3D Roughing (Mill3DRoughing)** — Z-level material removal around the 3D model in depth-per-pass steps.
- **3D Finishing (Mill3DFinishing)** — Surface-following finishing path with stepover (ball-nose recommended).
- **Plunge** — Vertical or ramp along path (RampAngleDeg); Direction: Climb / Conventional / Mixed; Order: DepthFirst / LevelFirst.

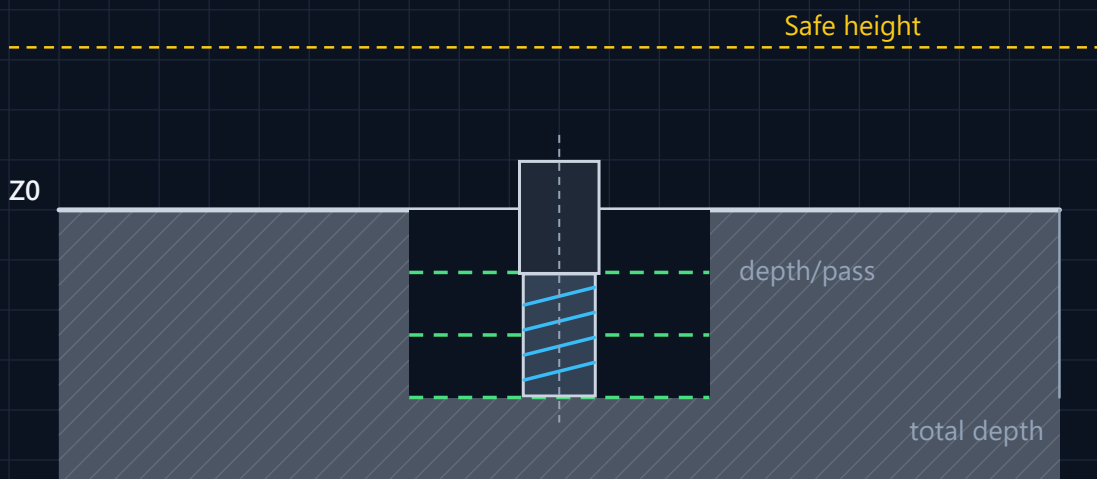
- **Tabs** — Profile TabCount, TabLength, TabHeight – the part does not come loose.

3D roughing and finishing

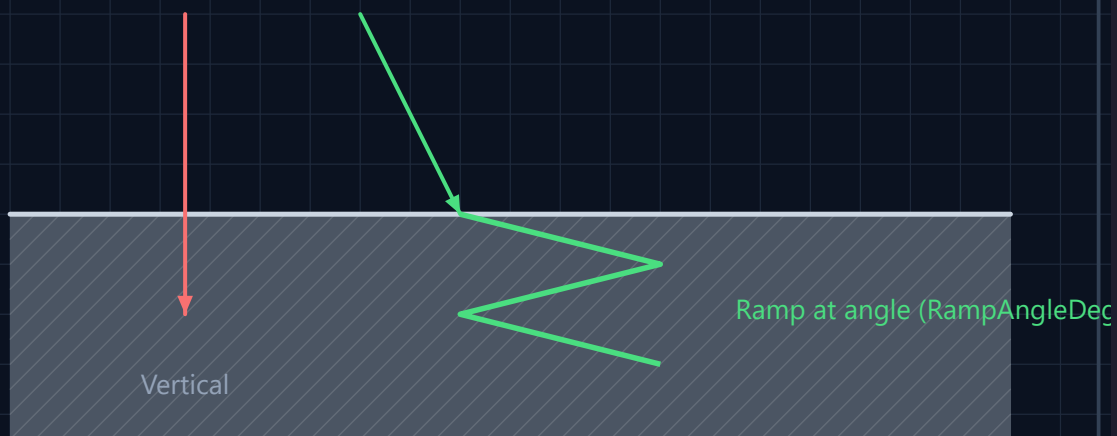


13.2 Common parameters

Depth, depth per pass, safe height



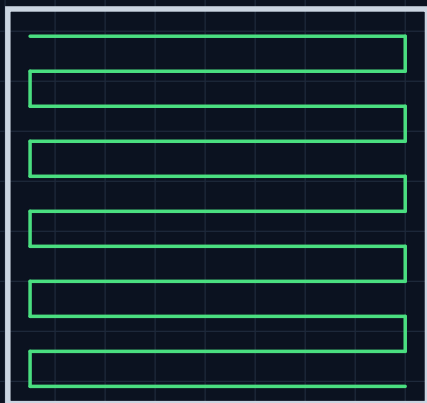
Plunge: vertical vs. ramp along path



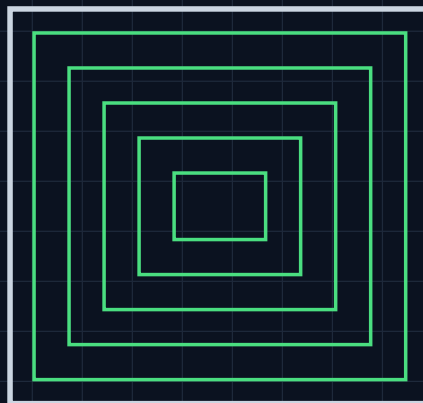
- **Tool** — Diameter, spindle speed, feed, plunge rate (from the tool library or manual entry)
- **Depth (mm)** — Total cutting depth
- **Depth per pass** — Depth of one pass (DepthPerPass)
- **Bizt. mag.** — Safe height (SafeHeight) — the tool rises here during rapid moves

13.3 Pocket milling in detail

Pocket strategies



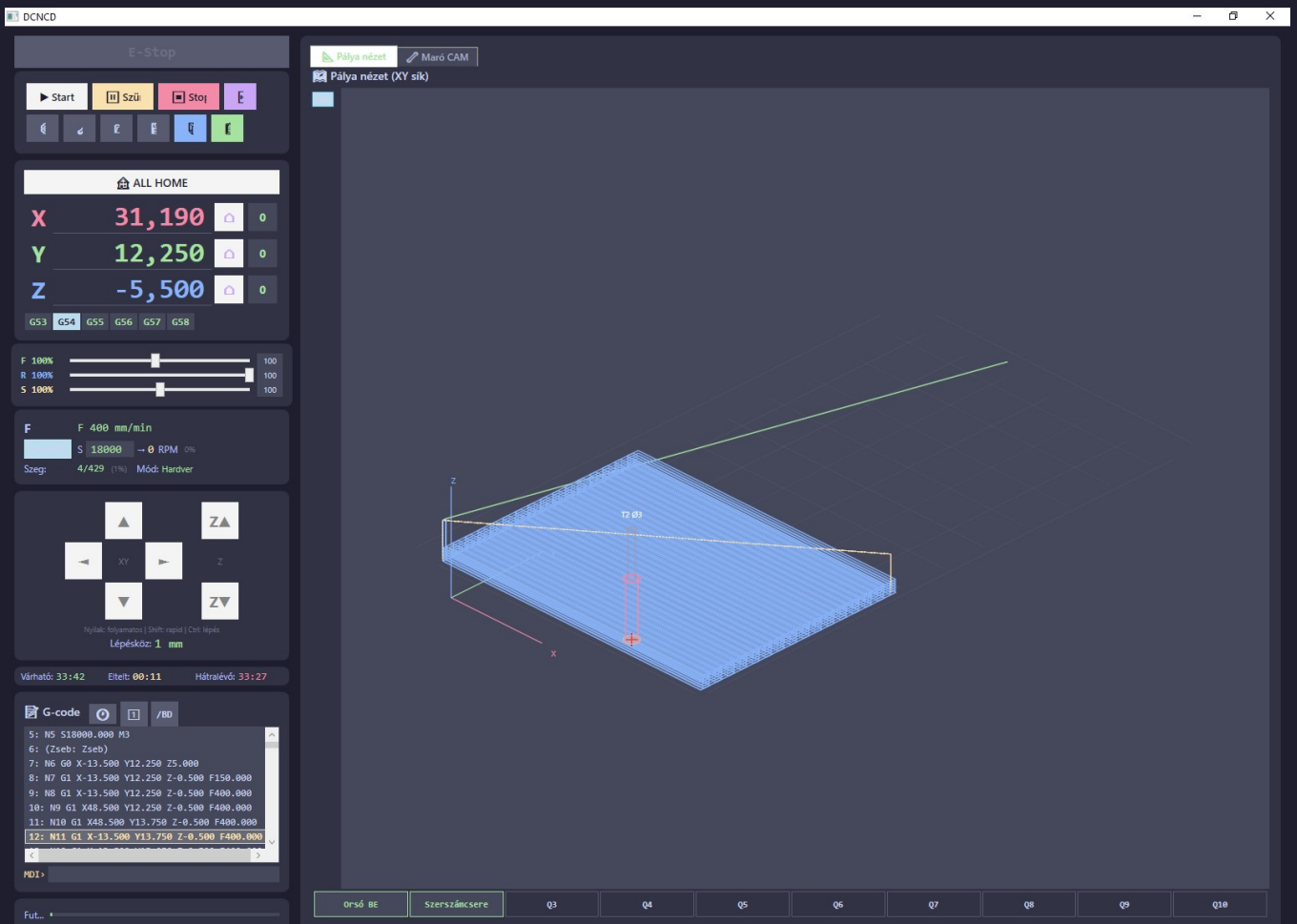
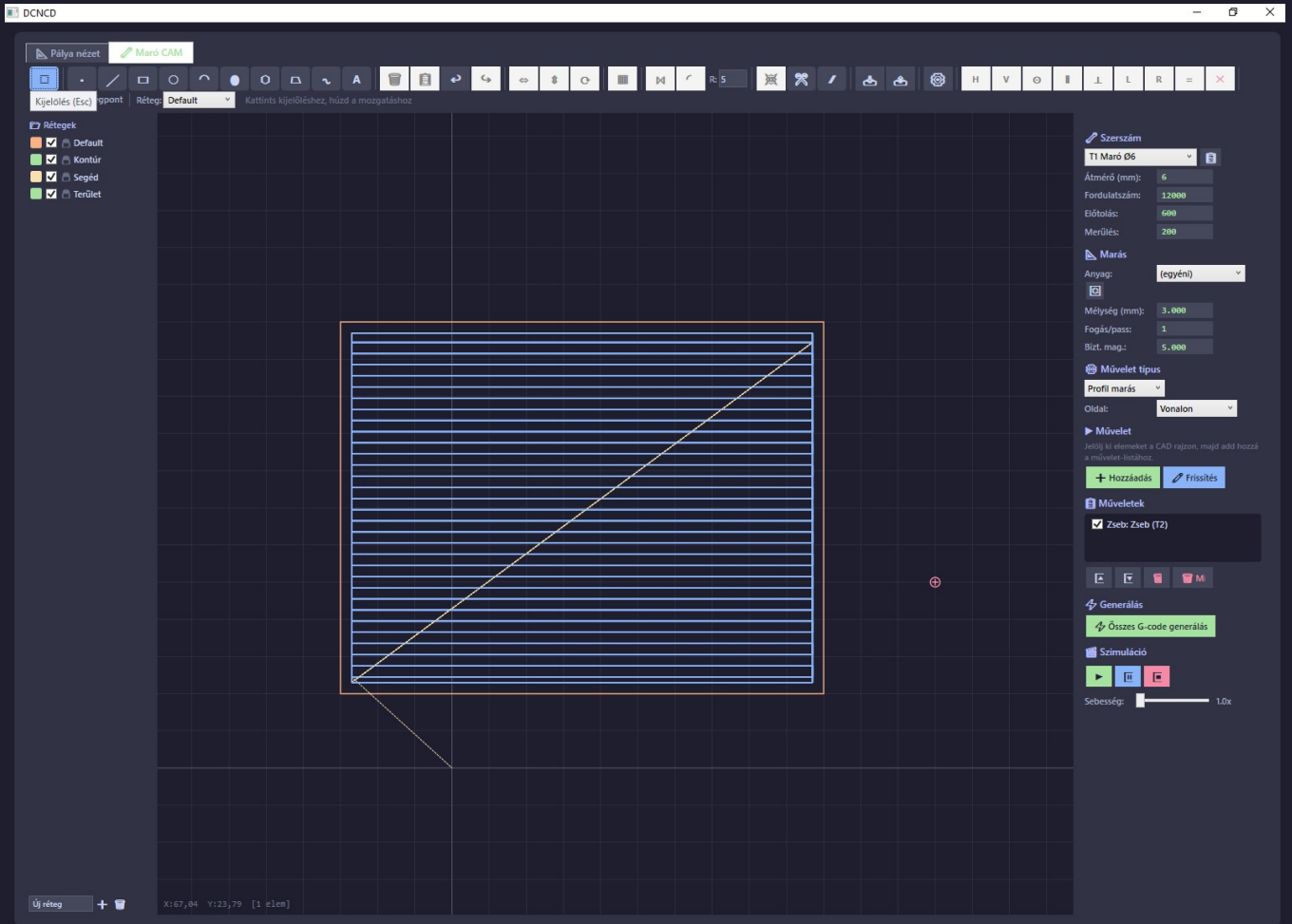
Parallel (angle, stepover %)

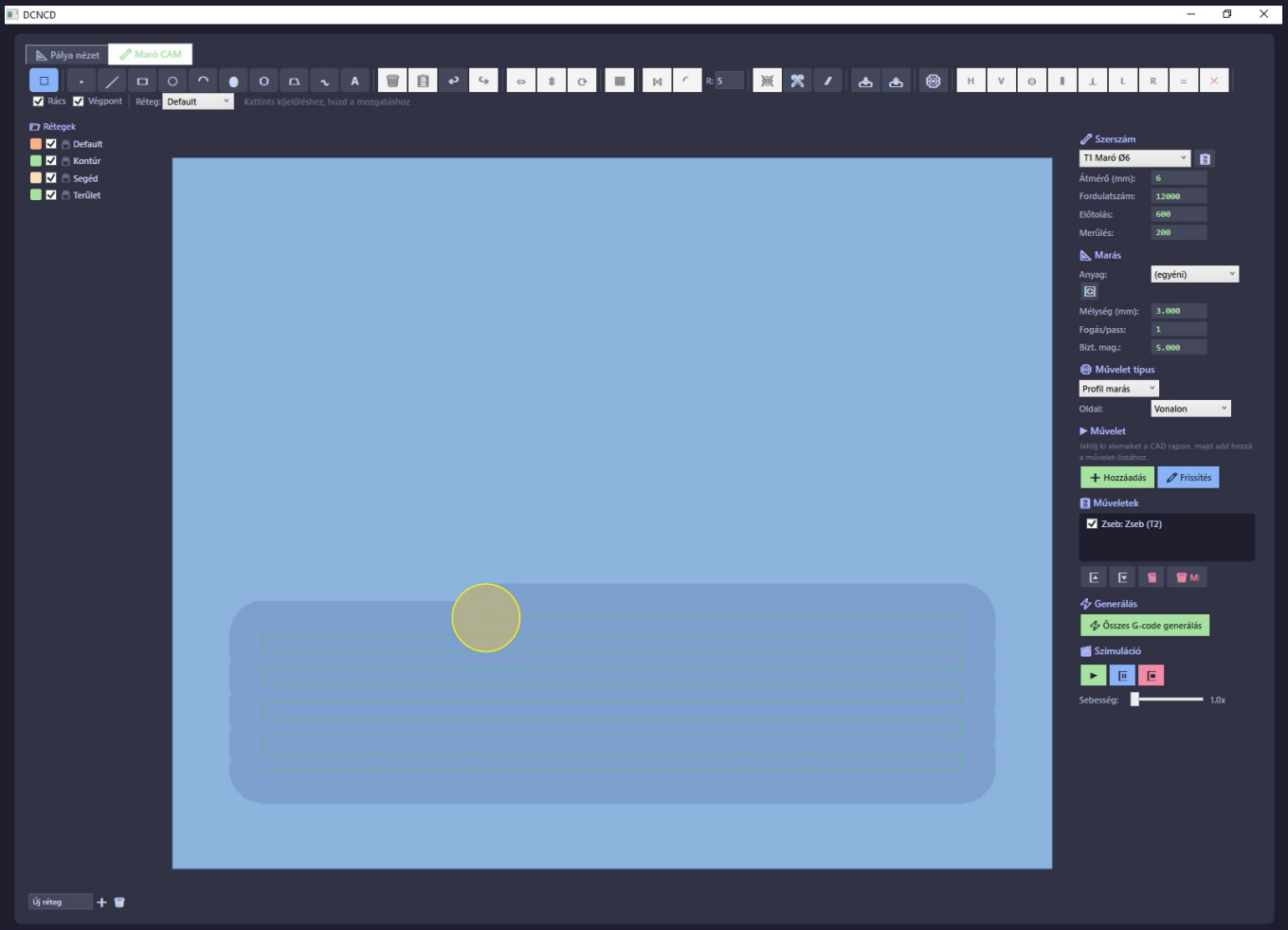


Concentric (+ finish contour)

Pocket milling removes material inside a closed contour. Two fill strategies are available:

- **Parallel (zigzag)** — Parallel scanlines at the given angle. Fast material removal on large areas.
- **Concentric** — Rings shrinking inward. Better surface quality, lower tool load.





Island support

If one or more closed contours are inside a pocket (e.g. circle, rectangle), the system automatically recognizes them as **islands**. The tool avoids the islands — it does not mill into them.

- **Usage:** Select all contours (outer + inner islands), choose the "Pocket milling" type, then click "Add".
- **Automatic detection:** The closed contour with the largest area becomes the pocket boundary; all smaller closed contours inside it → islands.
- **With the parallel strategy:** The scanlines are split at the islands — the tool jumps over with retract + rapid.
- **With the concentric strategy:** The ring points are skipped over the island areas.
- **Finishing contour:** If enabled, a finish pass runs along the pocket wall and around every island.

Region-based fill (parallel strategy)

When the pocket is not made of contiguous spans — e.g. U / horseshoe shape, ring, or an island splits it — the parallel fill **groups the scanlines into connected regions**: it mills one leg completely first, then continues with the next region (other leg, arc part) from the nearest entry point. The tool no longer jumps between the legs on every line, so retract + rapid moves are significantly reduced. Between regions the configured **Link move** (Straight / Arc / Retract) applies.

13.4 V-Carve machining

The V-Carve operation machines with a V-bit between two contours (left/right boundary). The depth is computed automatically from the local width and the V-bit angle:

$$\text{depth} = (\text{width} / 2) / \tan(\text{angle} / 2)$$

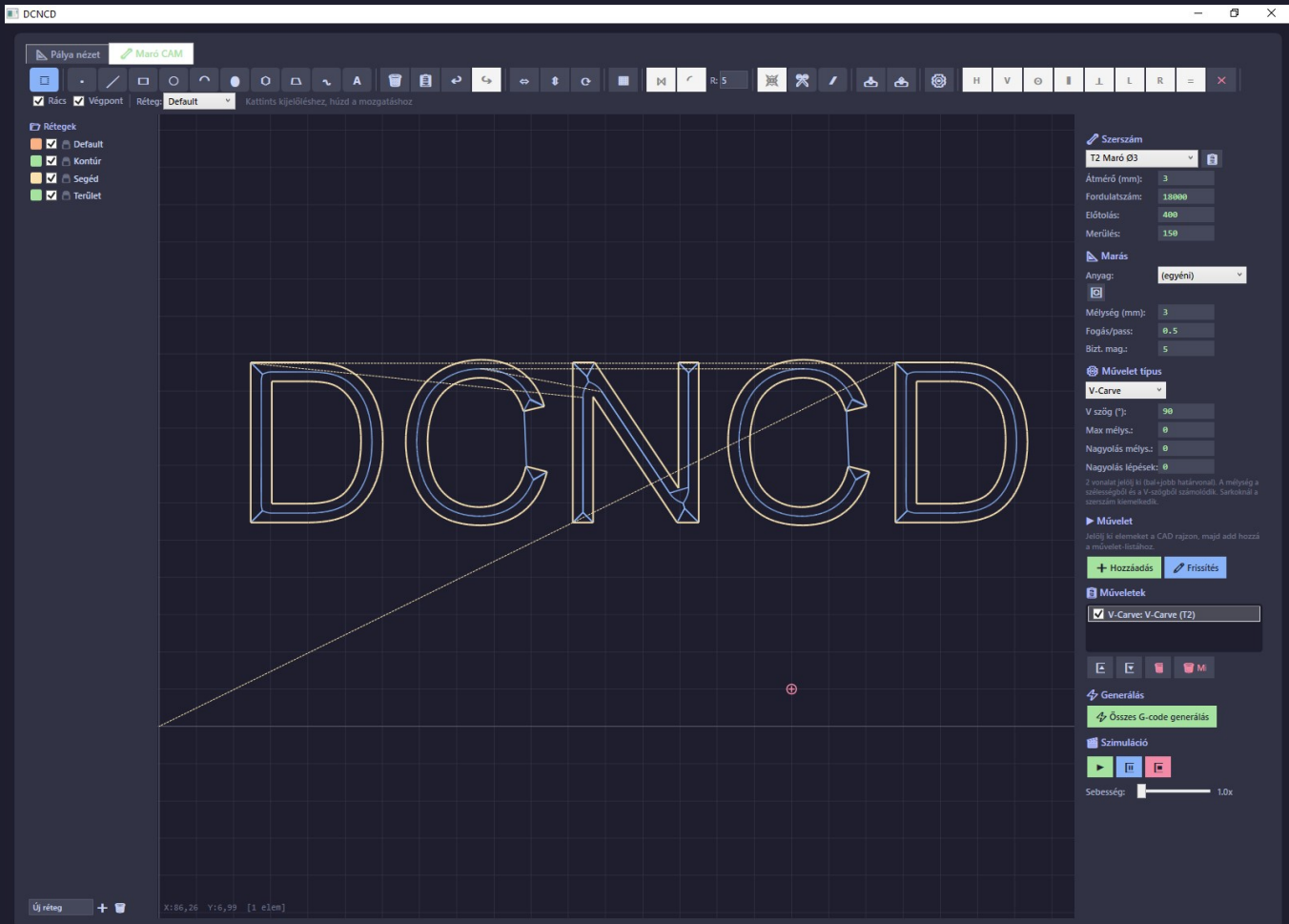
- **V angle (°)** — Included angle of the V-bit (e.g. 60°, 90°). Taken automatically from the tool library if a V-bit type tool is selected.
- **Max depth** — Maximum depth limit (0 = no limit, only geometry counts).
- **Corner handling** — Where the contours narrow (corner), the width decreases → the depth decreases automatically

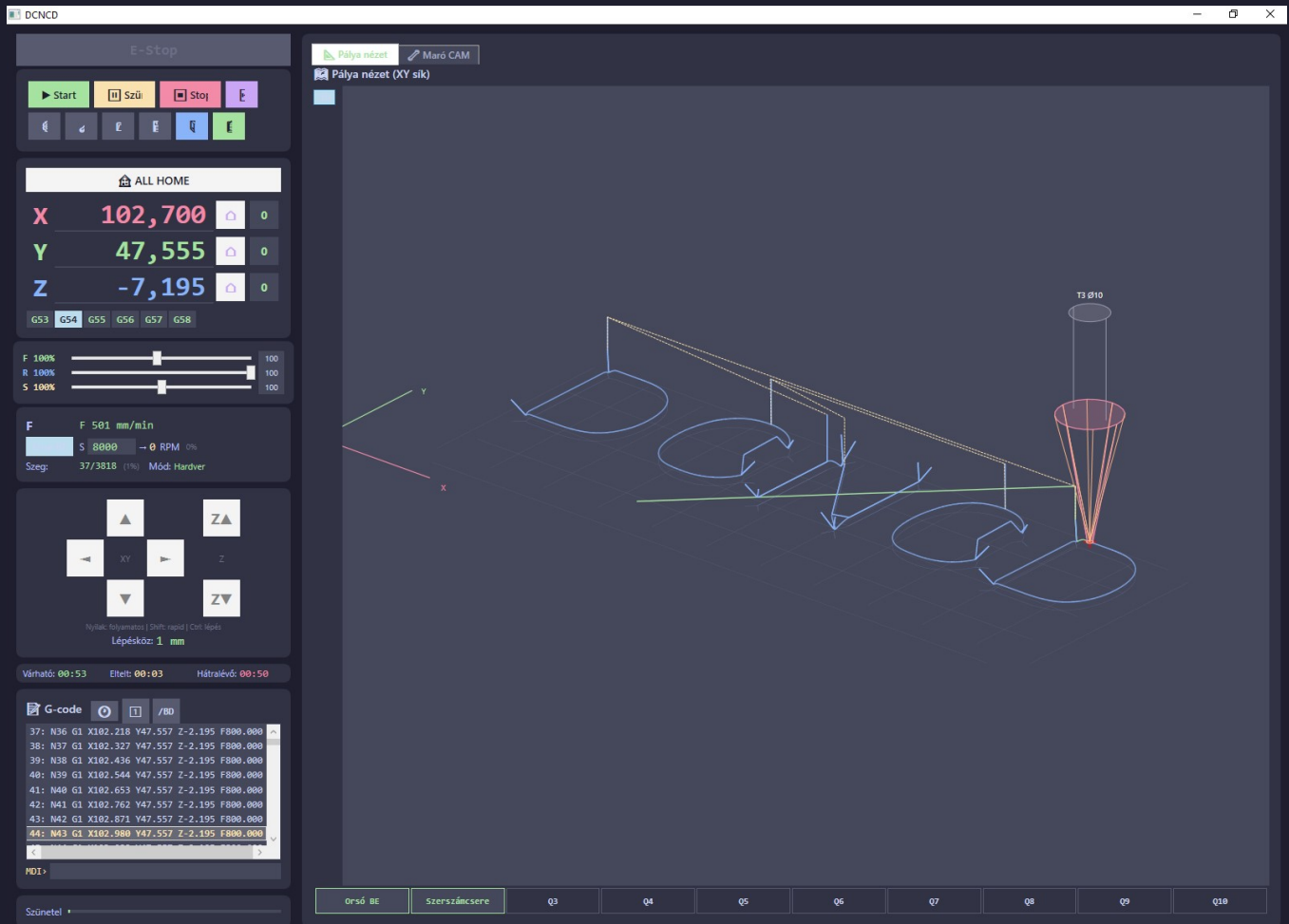
→ the tool lifts. This gives V-Carve its characteristic look.

- **Roughing (optional)** — If a roughing depth and step count are given, it first roughs at a flat depth (even with a flat end mill), then the V-bit does the finishing with the variable depth.

Usage

1. Draw two contours (left and right boundary) — or a single closed contour
2. Select both
3. Choose the "V-Carve" operation, set the V angle
4. Click "Add"





13.5 V-bit profile milling

A profil marás is támogatja a V-maró szerszámot. Ha a kiválasztott szerszám V-bit típusú ($VBitAngle > 0$), the toolpath offset is computed automatically from the depth and the V angle:

$$\text{offset} = \text{depth} \times \tan(\text{angle} / 2)$$

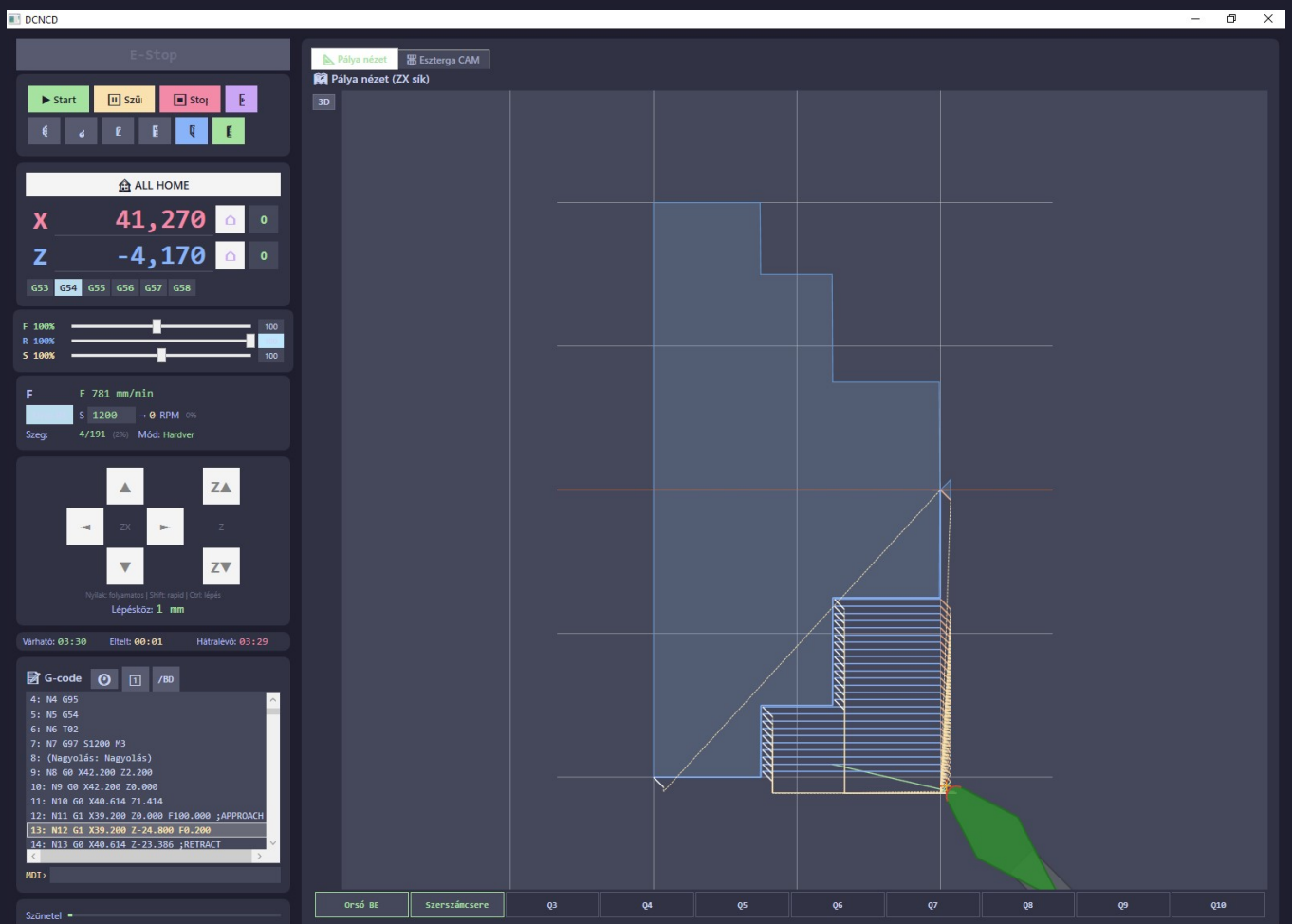
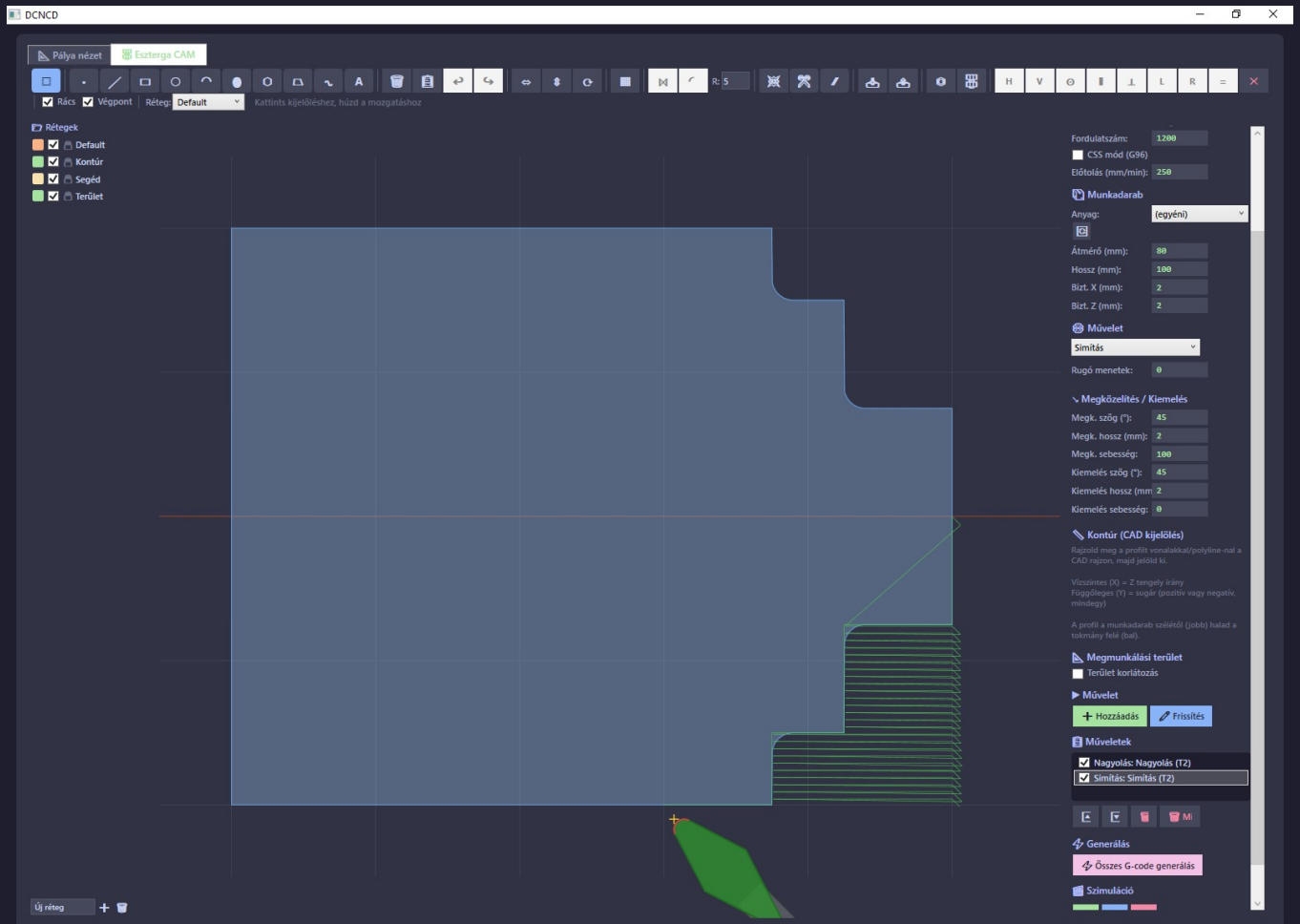
This means the V-bit runs with a different offset in each pass — going deeper → wider cut → larger offset. So both outside and inside it positions itself exactly where the given depth requires.

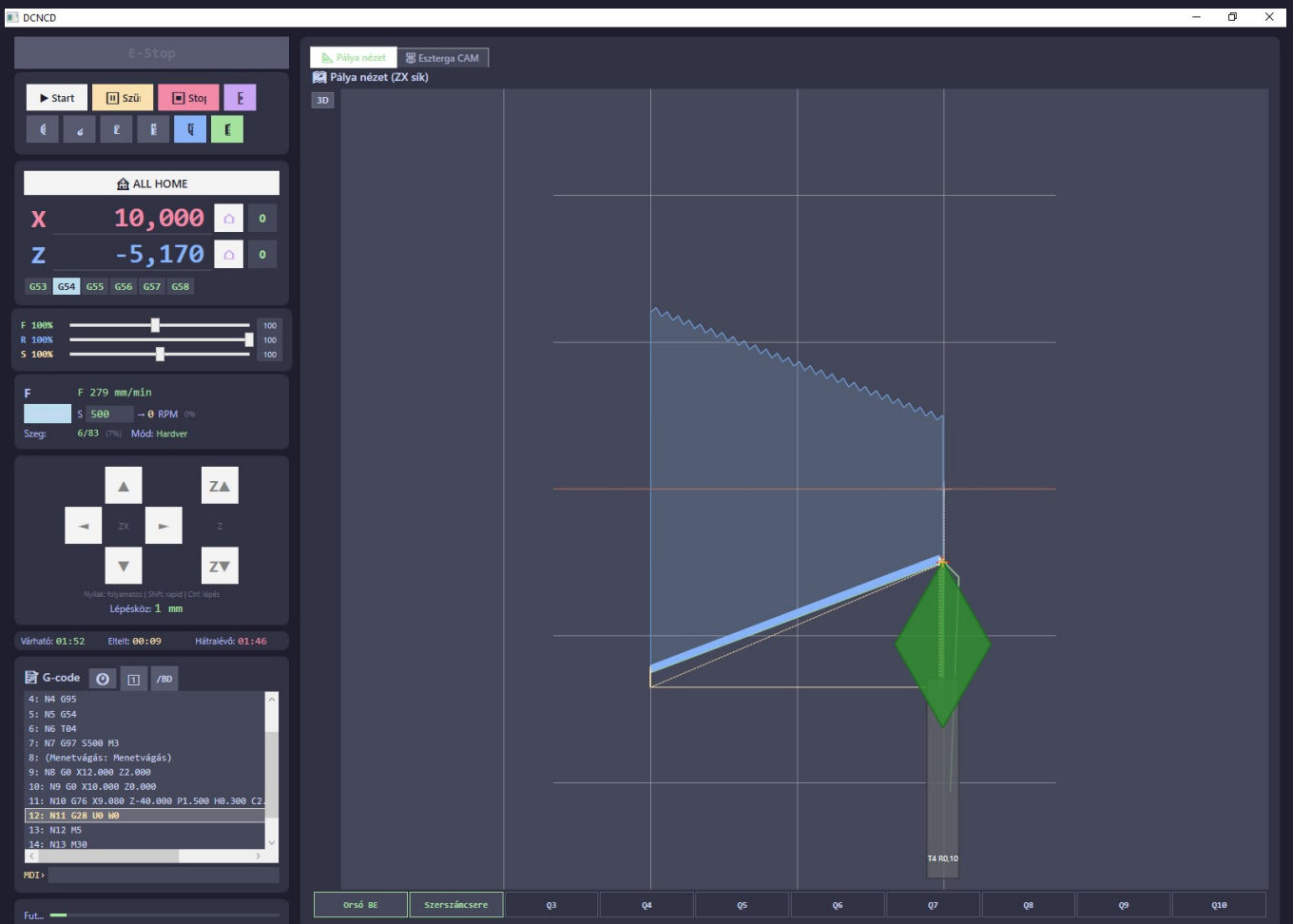
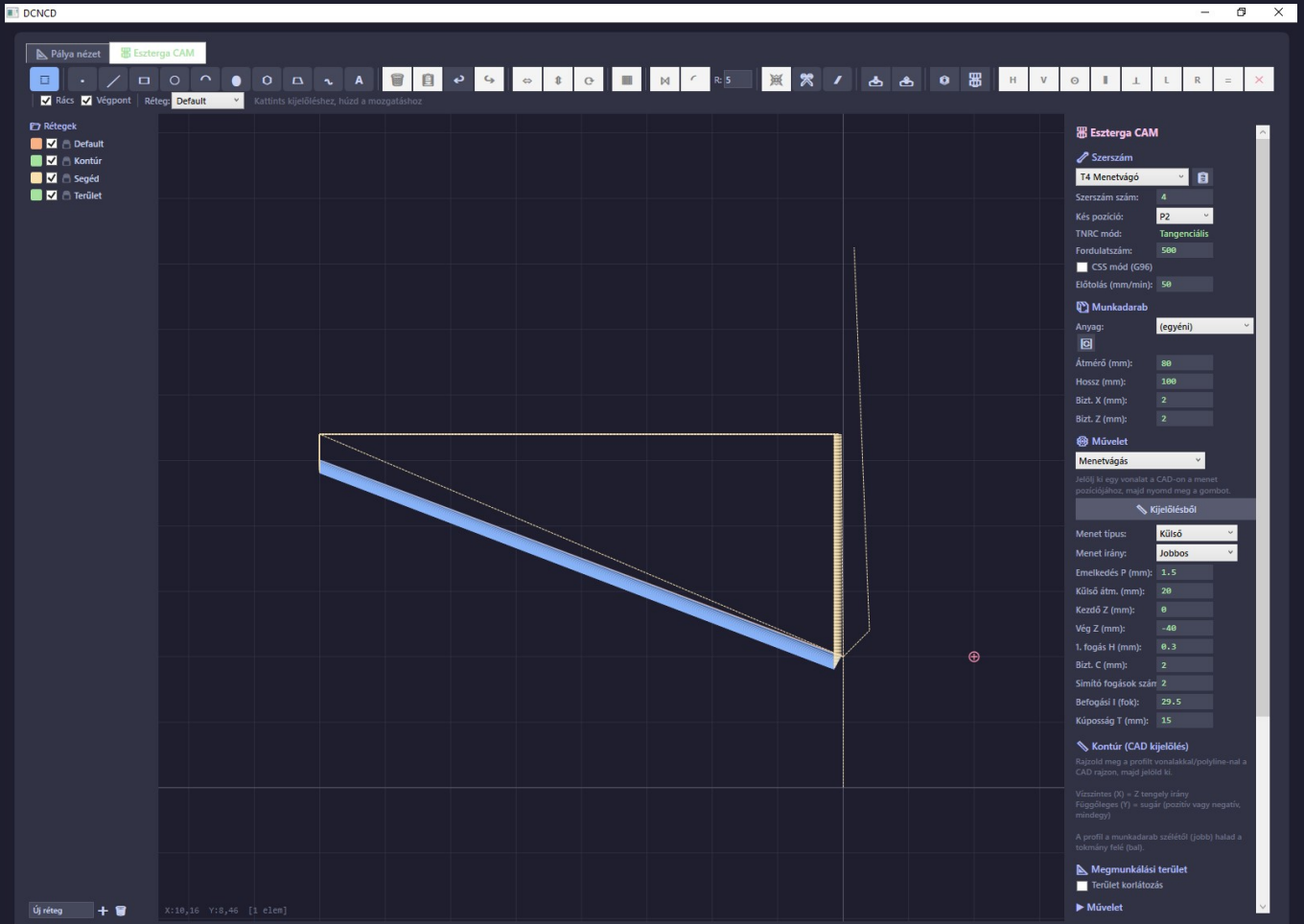
- Multi-step milling: recomputes the offset at each depth of cut
- Compatible: On line / Outside / Inside / Left / Right side setting

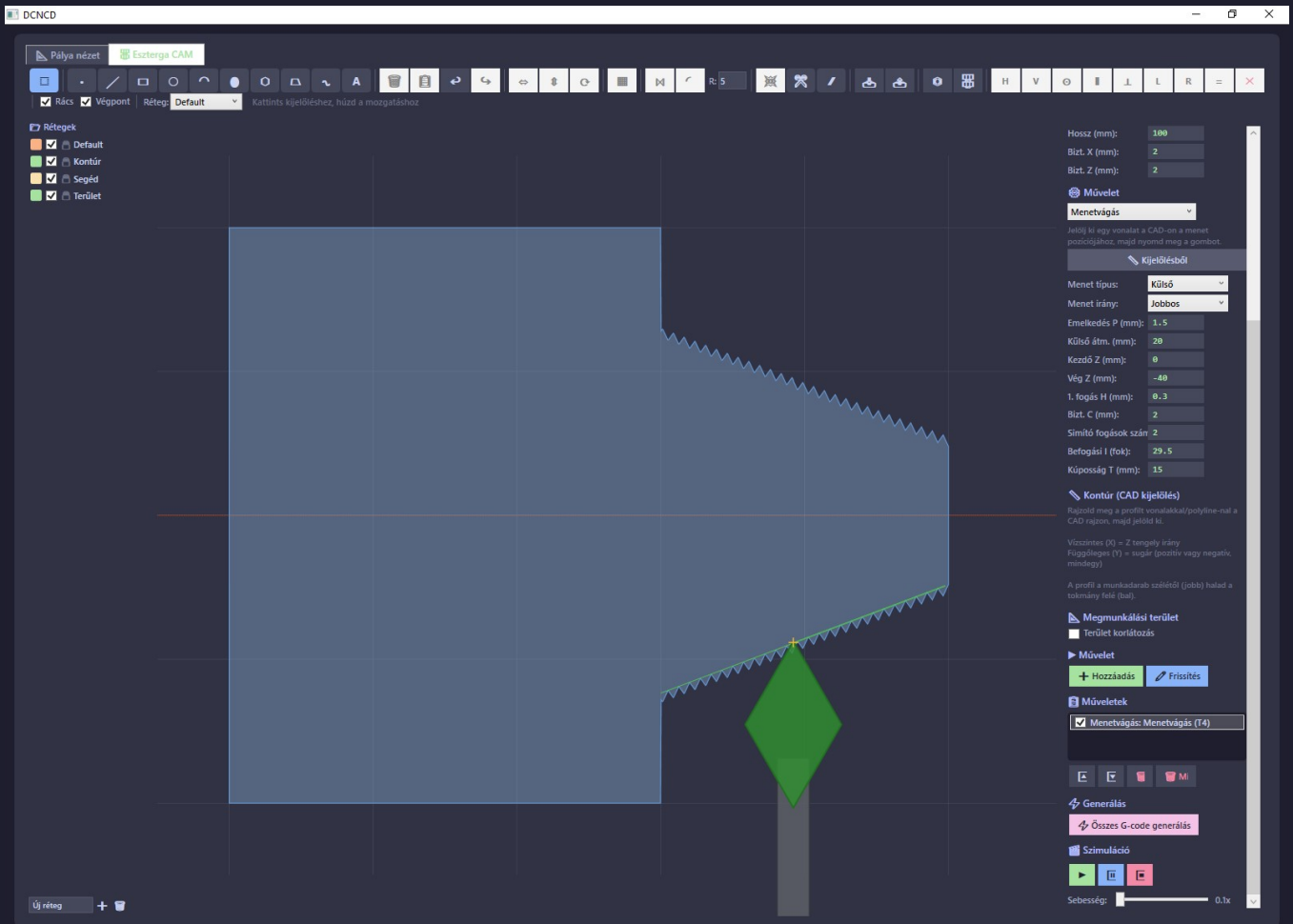
14. Lathe CAM

On the "Lathe CAM" tab you can draw the workpiece contour and generate lathe toolpaths. In the CAD drawing the horizontal axis (X) corresponds to the Z axis, the vertical (Y) to the radius (X). Several operations can be chained into a single G-code program.

Lathe CAM — Roughing and finishing:

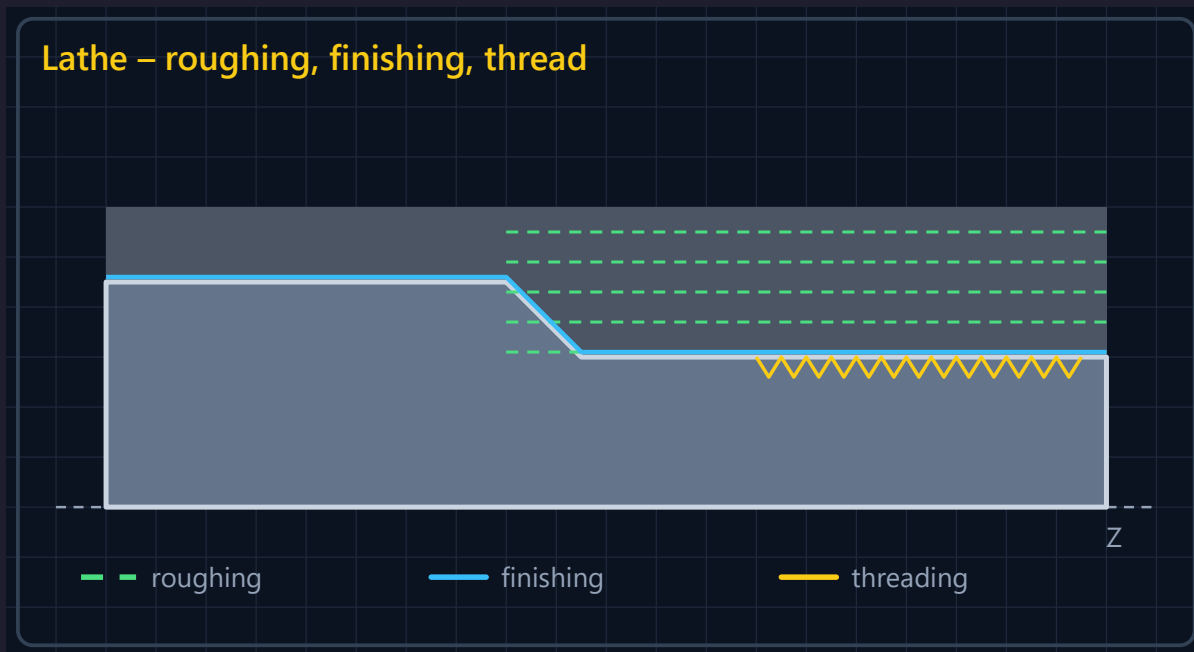






14.1 Operation types

Lathe – roughing, finishing, thread



Operation	Description
Roughing	External roughing along the contour with depth of cut and finishing allowance settings. Optional "Finish too" switch.
Finishing	Exact contour following to reach the final profile, spring passes option.
Threading	G76 cycle threading. External/internal, right/left-hand, lead, taper, infeed angle, spring pass configurable. Can also be parameterized from a CAD line.
Grooving	Grooving tool at a given Z position, down to the target diameter, with peck depth.

Facing	Turning the face with radial passes.
Drilling	Center drilling / drilling on the axis, with peck depth.
Internal roughing	Internal turning (boring): roughs from the pre-drilled diameter to the contour. Optiona "Finish too" switch.
Internal finishing	Exact finishing of the internal contour, with spring passes option.

14.2 Approach / Retract

For lathe operations the tool approach and retract mode can be configured:

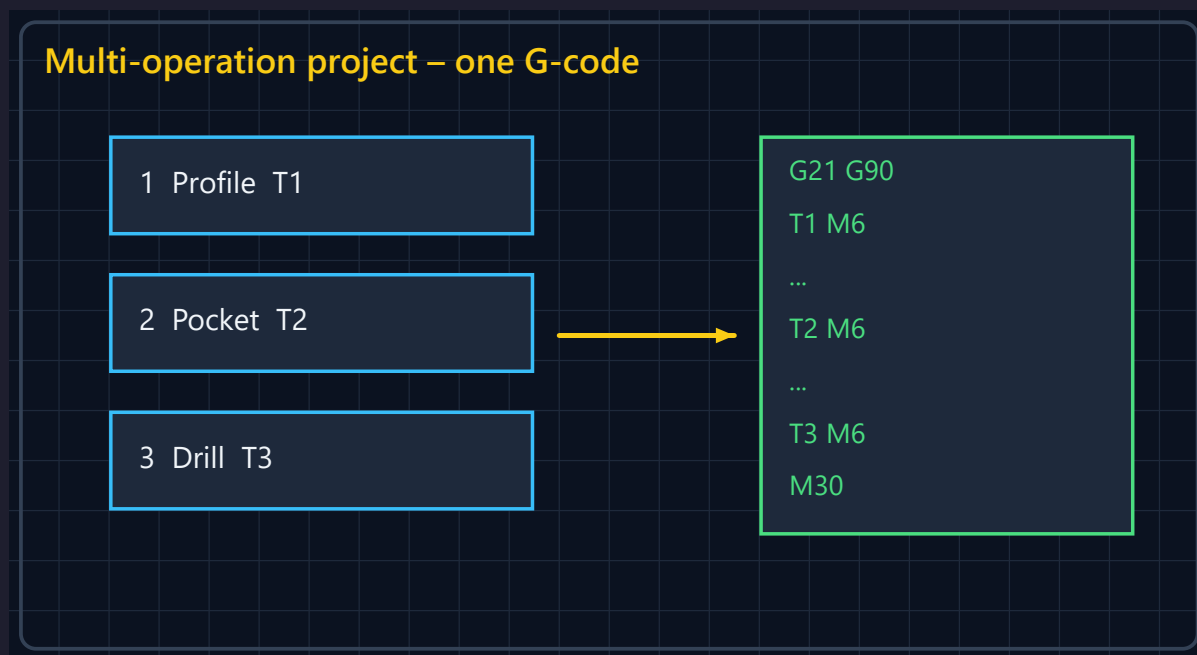
- **Angle (°)** — Approach / retract angle (0° = straight X plunge, 45° = diagonal, 90° = Z direction)
- **Hossz (mm)** — Length of the approach / retract move
- **Speed** — Feed for approach / retract (0 = rapid)

14.3 Machining region

Optionally the machining can be restricted to a defined region:

- **Region limit ✓** — Set X min/max (radius), Z min/max values
-  **From selection** — Select a rectangle in the CAD drawing and it automatically sets the region boundaries
- The system places the selected region on the "Terület" layer and updates it automatically when editing

15. Multi-operation project (Multi-toolpath)



Both the mill and the lathe CAM support chaining multiple operations into a single G-code program. This allows planning complex machining (e.g. drilling → internal roughing → internal finishing → external roughing → finishing → threading) executed in a single run.

15.1 Usage

1. Choose an operation type and set the parameters on the panel
2. Select geometry in the CAD drawing (if needed)
3. Click the **+** **Add** button — the operation is added to the list
4. Repeat with other operation types and tools
5. Click the  **Generate all G-code** button — all enabled operations go into a single G-code

15.2 Operation list management

- ✓ **Checkbox** — Enable/disable the operation (a disabled operation is not included in the G-code)
- ▲ ▼ — Move the operation up/down in the order
- ✎ **Update** — Overwrite the selected operation with the current panel settings
- 🗑 **Delete** — Remove the selected operation
- 🗑 **Mind** — Delete all operations

When generating G-code the application handles tool changes (T codes) automatically: if consecutive operations use the same tool, the change is not repeated. The header and footer (G18/G17, G21, G54, M5, M30) are placed once at the start/end of the program.

16. Simulation mode

16.1 What is it and why is it needed?

Most modern computers have no LPT (parallel) port, so the machine's control electronics cannot be driven directly. In that case the application automatically enters **simulation mode** . In this mode the whole software works the same — you can edit the G-code, run it, watch the toolpath — only the **Physical outputs** (step pulses, spindle, coolant) are simulated.

💡 Imagine: "All the machine's motors are unplugged from the wall, but the software runs perfectly — like a flight simulator!"

16.2 How is it activated?

At startup the application tries to open the LPT port driver. If it is not available (not installed, USB-LPT adapter not connected, or Windows does not allow it), it automatically enters simulation mode. You do not need to configure anything!

Helyzet	What happens?
LPT driver available	Normal mode — step pulses go to the port, the machine moves
LPT driver NOT available	Simulation mode — everything is simulated, indicated in the bottom-left corner

⚠ Always visible in the bottom-left corner: "🖨 SIMULATION" label when you are in simulated mode!

16.2.1 Second LPT port (dual port)

If your machine needs more inputs/outputs than a single LPT port provides (8 Data + 4 Control outputs, 5 Status inputs), you can set the **2. LPT port** address in Settings. The full pin set of the second port is available:

Pin range	Regiszter	Usage
20–27	2nd port Data (output)	Extra step/dir, spindle, coolant, PWM, etc.
30–33	2nd port Control (output)	4 extra outputs (HW inverted: bit 0,1,3)
20–27	2nd port Status (input)	Extra limit, home, probe, encoder

Typical addresses: **0x378** (LPT1), **0x278** (LPT2), **0x3BC** (LPT3). 2nd port address = 0 → disabled (single port mode).

16.2.2 LPT card selection — why does the hardware matter?

For precise timing of the step pulses it matters which LPT card you use. The software side of the application compensates every known timing error (rising-edge latching, minimum LOW time, backpressure handshake, real-time priority), but if the port write itself has **high and unpredictable latency** (jitter), then during a long run (10+ min, high step rate, lathe / threading) step loss can occur.

Card type				Rating	Note
Brainboxes (UC-475, PX-157, PCI-1L LPT...)				✓ RECOMMENDED	Dedicated, low latency, jitter-free. Suitable for long-run lat/ threading operation too.
Onboard	SuperIO	LPT	(old	⚠ LIMITED	Fine for short runs, but drift may occur on long threading. Ch

machines)		with the Latency benchmark.
Sunix / MosChip PCI(e) LPT	✗ AVOID	High jitter in compatibility mode, step loss on long runs. Typ port address: 0xD000–0xEFFF.
USB → LPT "adapter"	✗ NOT USABLE	Not a real LPT — only for the printer protocol, no direct port Unsuitable for CNC.

16.2.3 Latency benchmark — how to interpret it?

A **Settings** → **LPT base address** section, the  **Latency benchmark** button measures the time of 2000 consecutive port writes and prints the min / average / P99 / max / jitter values in nanoseconds, plus a rating.

Rating	Average (ns)	P99 (ns)	Jitter (ns)
✓ EXCELLENT	< 1 500	< 4 000	< 20 000
⚠ ACCEPTABLE	< 4 000	< 15 000	< 80 000
✗ GYENGE	> 4 000	> 15 000	> 80 000

💡 A **jitter** (max – min) is the most important indicator: this causes long-run drift, not the average. A card with 800 ns average but 200 000 ns max spikes is worse than one with 1 500 ns average but 5 000 ns spikes.

Note: the benchmark gives a representative picture when run on an idle machine, BEFORE starting the CNC run. Background processes (antivirus, Windows Update) can temporarily worsen the value.

16.3 What works in simulation?

Function	Works?	Note
G-code editing	✓ Igen	Full editor, load, save
Toolpath display	✓ Igen	2D toolpath view, rapid/cut colors
CAD drawing	✓ Igen	All drawing tools available
CAM generation	✓ Igen	Mill and lathe CAM, G-code generation
Program run	✓ Igen	DRO updates, current line highlighted, toolpath drawn
Macros (M6, custom)	✓ Igen	MSG, WAIT, motions run simulated
3D simulation	✓ Igen	Mill: top-view material removal, Lathe: cross-section
Physical outputs	✗ No	Step pulses, spindle, coolant do not reach the port

16.4 Timing and speed

In simulation mode the **timings are real** — acceleration, feed, rapid speed and dwell (G4) wait take exactly as long as on a real machine. This matters because the run time can be estimated realistically.

 **Tip:** If you only want to look at the toolpath (time does not matter), speed up the simulation with the speed slider!

16.5 Visual simulation (material removal)

The simulation view shows real-time material removal — you can see the tool "carving" the material:

Mode	What do you see?
 Mill simulation	Top view (XY plane) — the workpiece is seen "from above", the cutter leaves a circular trace Depth is indicated by color shade.
 Lathe simulation	Cross-section view (ZX plane) — the tool cuts material off the cylinder. The chuck and the tool nose are also visible.

16.6 Simulation control buttons

Gomb	Function
▶ Play	Start simulation — the tool starts moving along the toolpath
⏏ Pause	Pause simulation — the tool stops at the current position
■ Stop	Stop simulation and reset to the start point

16.7 What to use it for?

1. **Program verification** — Run it in simulation before cutting real material! Check the toolpath, the timing, the tool changes.
2. **CAM development** — Experiment with the CAM parameters (depth, stepover, strategy) and see the result immediately.
3. **Learning** — Learn G-code without a real machine! Type the command into the MDI and watch what it does.
4. **Demonstration** — Show the planned machining to a customer or colleague with the simulation.

🔗 Bottom line: simulation mode is not a "limited version" — it is the full application, just without a physical machine! Ideal for planning, learning and verification.

17. M-code macro system

17.1 What is it and why is it needed?

In G-code the M-codes are "machine instructions" — for example M3 = spindle on, M5 = spindle off. But what if you want **custom behaviour** ? For example, at a tool change the machine should rise automatically, print "Change the tool!", then wait until you press Start? That is what the **macro system** is for — you can write your own script for any M-code!

💡 Imagine: "M6 is a button on the machine. Press it → the macro runs like a little robot: moves, writes a message, waits for you."

17.2 How do I open the editor?

1. Click the ⚙️ **Settings** button in the main window
2. Scroll down to the **Macros** section
3. Click the "**Macro editor**" gombra

The macro editor window opens; the list on the left shows all macros, on the right you can edit the script of the selected macro.

📁 Storage: %APPDATA%\CncController\Macros\macros.json — saved automatically.

17.3 Default M6 tool change

The application ships with a ready-made M6 macro. In G-code **T2 M6** means: "Select tool 2 and perform the change." The macro step by step:

```
; ===== M6 Tool change macro =====
IF #tool_req == #tool ; 🔍 Same tool?
LOG Same tool, change skipped
RETURN ; ⬅️ Exit – no change
ENDIF
SPINDLE_OFF ; 1🛑 Spindle stop
COOLANT_OFF ; 2🛑 Coolant off
RAPID Z50 ; 3📏 Raise Z to safe height
RAPID X0 Y0 ; 4📏 Move to change position
MSG Change the tool: T#tool_req – then press Start
WAIT ; 5🕒 Wait for Start (or Alt+R)
SET #tool = #tool_req ; 6🔄 Update the current tool number
LOG Tool change done: T#tool
```

🔗 So if you write T2 M6 in G-code: spindle stops → Z rises → XY goes to the change position → prints "Change T2" → waits → updates. Fully customizable!

17.4 Macro commands — full reference

Parancs	What does it do?
RAPID X.. Y.. Z..	Rapid move to the given position. Axis not written = does not move.
MOVE X.. Y.. Z.. F..	Feed move at the given speed (F = mm/min). Like G1.
SPINDLE_ON [CW CCW] S..	Spindle on. CW = clockwise, CCW = counter-clockwise. S = speed.
SPINDLE_OFF	Spindle stop (same as M5).
COOLANT_ON	Coolant on (same as M8).
COOLANT_OFF	Coolant off (same as M9).
DWELL P..	Wait P seconds. E.g.: DWELL P2 = 2 second pause.
WAIT	Stop — waits until the user presses Start (or Alt+R).
MSG text	Show a message in the status bar. Variables are substituted!
LOG text	Log entry into the log window. Ideal for debugging.
SET #var = value	Set a variable's value. E.g.: SET #tool = #tool_req
IF #var == value	Start of a conditional block. Operators: ==, !=, >, <, >=, <=
ENDIF	End of the conditional block. Mandatory for every IF!
RETURN	Immediate macro exit. Useful in conditional cases (e.g. no change).
PROBE X.. Y.. Z.. F..	Probing move — moves until touch. Result: #probe_tripped, #probe_x/y/z (see 26.7).
CALL file [L..]	Run an external G-code subprogram file from the macro, L = repeat count (see 17.10).
M98 (file) [L..]	Same as CALL — Fanuc-style syntax.

⚠ Lines starting with a semicolon (;) are comments — the system skips them. Use them generously!

17.5 Variables — where does the data come from?

In macros you can use variables starting with the # sign. The system fills the built-in variables automatically with the machine's current state:

Variable	Meaning	Example value
#x	Current X position	25.400 (mm)
#y	Current Y position	10.000 (mm)
#z	Current Z position	50.000 (mm)
#tool	Current tool number	1 (T1 is loaded)
#tool_req	Requested tool (T word)	2 (G-code requested T2)
#feed	Current feed	200 (mm/min)
#rpm	Spindle speed	12000 (rev/min)
#1..#99	User variables	Store anything (number). Default: 0

💡 Variables are also substituted in MSG and LOG commands! E.g.: MSG Tool: T#tool → prints: "Tool: T1"

17.6 Creating a custom macro — step by step

1. Open the **Macro editor** (Settings → Macros → Macro editor)
2. Click the **+** button to add a new macro
3. Enter the **M-code number** (e.g. 600 → callable as M600 from G-code)
4. Enter the **nevet** (e.g. "Tool length measurement") — this only helps you remember
5. Write the **script-et** into the editor on the right
6. Kapcsold **the enabled** checkbox (✓)
7. Close the editor — it is saved automatically

After that, in G-code **M600** calls the macro!

17.7 Full example — custom "workpiece measurement" macro (M601)

This macro raises Z, goes to the measuring point and waits for the user:

```
; ===== M601 Workpiece measuring point =====  
SPINDLE_OFF ;  Spindle off (safety!)  
RAPID Z30 ;  Raise Z  
RAPID X50 Y50 ;  Move to the measuring point  
MSG Measure the workpiece, then press Start!  
WAIT ;  Wait for the user  
LOG Measurement done, position: X#x Y#y Z#z  
MSG Measurement recorded – program continues...  
DWELL P1 ;  1 s pause (so it can be read)
```

 In G-code you only write: M601 → and the whole measuring routine runs!

17.8 Conditional logic — IF / ENDIF example

In macros you can make commands conditional. Example: turn on coolant only if the spindle is running:

```
IF #rpm > 0 ;  Is the spindle running?  
COOLANT_ON ;  Yes → coolant on  
MSG Coolant on  
ENDIF
```

Supported operators:

Operator	Meaning	Example
==	Equal	IF #tool == 1
!=	Not equal	IF #tool_req != #tool
>	Nagyobb	IF #rpm > 0
<	Kisebb	IF #z < 5
>=	Greater or equal	IF #feed >= 100
<=	Less or equal	IF #1 <= 10

17.9 Important rules and tips

Rule	Details
Comments	Lines starting with a semicolon (;). Use them generously for documentation!
One macro = one M-code	Each M-code has one macro. If you need M6 + M600, that is two separate macros.
Enabled/Disabled	With the  checkbox you can switch it on/off without deleting the code.
WAIT + simulation	Works in simulation mode too! It waits for the Start button (Alt+R).
Variables in MSG	In MSG and LOG the #variables are automatically replaced with the actual value.
RETURN = immediate exit	The macro runs to the end anyway — RETURN is only needed for early exit.

 Tip: Start simple! Write a MSG + WAIT macro, see it run, then extend it with motions and conditions.

17.10 Running an external G-code file from a macro (CALL / M98)

From a macro you can also run a **complete G-code file** — e.g. a tool measuring routine or a repeating machining pattern. The file goes through the standard G-code processor (subprograms work in it too), then the motions are executed on the macro channel.

```
; Macro example:  
MSG Tool measurement starting...  
CALL toolmeasure.tap ; run external file once  
M98 (drill_pattern) L3 ; Fanuc style, repeated 3x  
MSG Done!
```

- File lookup: the folder of the loaded program, then **%APPDATA%\CncController\Subprograms**
- For a name without extension it automatically tries: .tap, .nc, .ngc, .gcode, .txt
- The log shows: "CALL filename L1 — N segments" — if the file is not found, an error message appears
- An external file can also be called from a G-code program and from MDI — see chapter 23.7

18. Tool table (Tool Offset Table)

18.1 What is it and why is it needed?

Imagine you have 5 different tools in the workshop — a short end mill, a long drill, etc. Each has a different length, so when you swap them the machine's "Z zero" changes. The **tool table** stores how much longer or shorter each tool is than the others. So the machine corrects the difference automatically and the workpiece always comes out to size.

💡 Simply: "I tell the machine that tool 2 is 3mm longer than tool 1 — and the machine automatically goes 3mm lower."

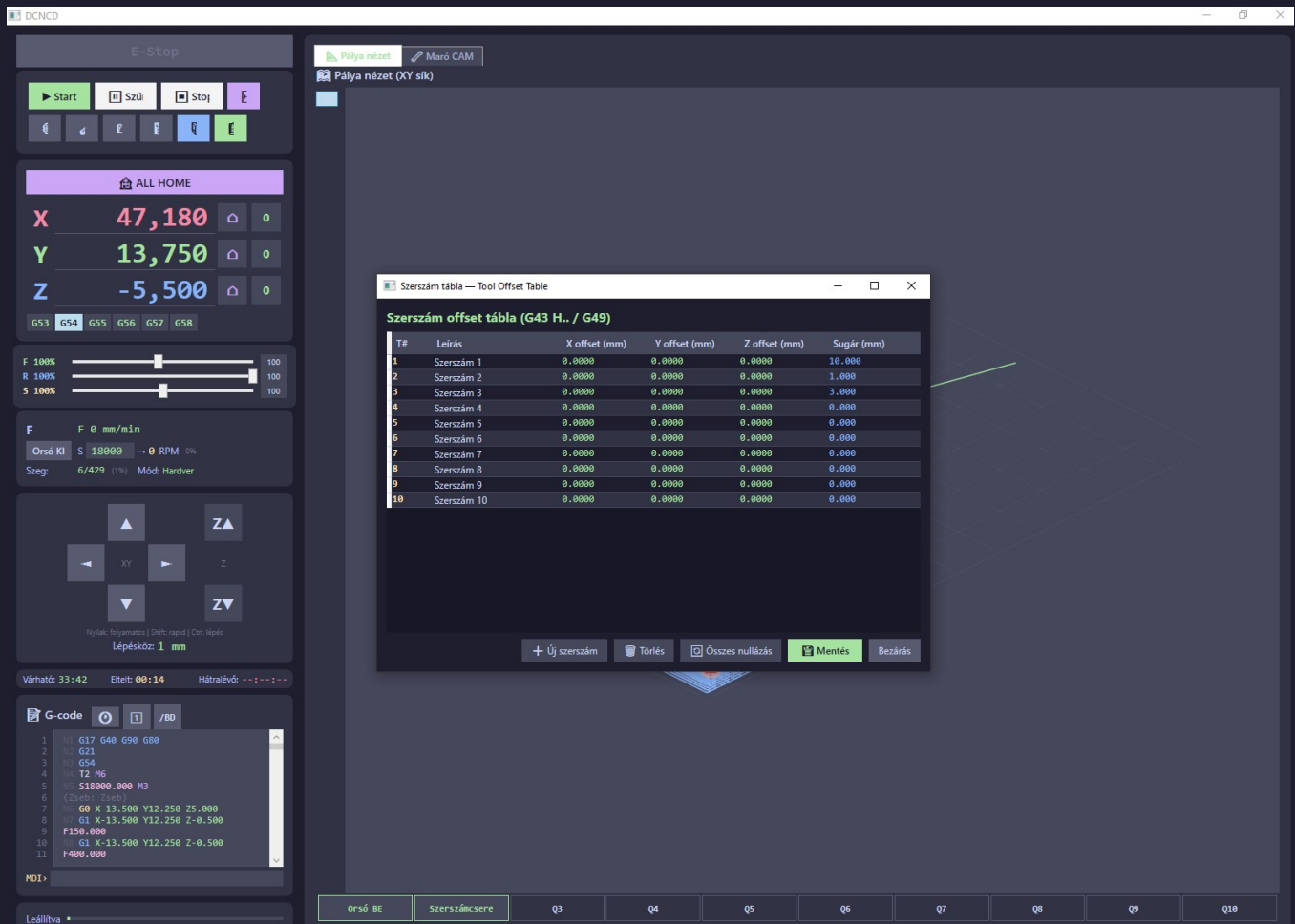
18.2 How do I open it?

At the top of the main window click the  button (the wrench icon next to the  settings). A table opens where you can fill in the tools:

T#	Description	X offset	Y offset	Z offset	Radius
T1	Mill Ø6	0.000	0.000	-52.345	3.000
T2	Drill Ø5	0.000	0.000	-55.120	2.500

- **T#** — Tool number (the same number as in G-code: T1, T2, ...)
- **Description** — Write anything here, it helps you remember which is which
- **X / Y / Z offset** — How much to shift the position in mm. In most cases only Z is used (tool length)
- **Radius** — Tool radius in mm (for G41/G42 radius compensation — see chapter 21)

📁 The table is saved automatically to: **%APPDATA%\CncController\tooltable.json**



The screenshot shows the Maro CAM software interface. On the left is a control panel with buttons for Start, Stop, and various axis movements. The main area displays the current position: X=47,180, Y=13,750, Z=-5,500. A 'Szerszám offset tábla (G43 H.. / G49)' window is open, showing a table with 10 tool entries. The table has columns for T#, Leírás, X offset (mm), Y offset (mm), Z offset (mm), and Sugár (mm). All offsets are currently 0.000, and all radii are 0.000. The bottom of the interface shows a G-code editor with a program listing and a status bar at the bottom with buttons for Orsó BE, Szerszámcseré, and various tool numbers (Q3-Q10).

T#	Leírás	X offset (mm)	Y offset (mm)	Z offset (mm)	Sugár (mm)
1	Szerszám 1	0.0000	0.0000	0.0000	10.000
2	Szerszám 2	0.0000	0.0000	0.0000	1.000
3	Szerszám 3	0.0000	0.0000	0.0000	3.000
4	Szerszám 4	0.0000	0.0000	0.0000	0.000
5	Szerszám 5	0.0000	0.0000	0.0000	0.000
6	Szerszám 6	0.0000	0.0000	0.0000	0.000
7	Szerszám 7	0.0000	0.0000	0.0000	0.000
8	Szerszám 8	0.0000	0.0000	0.0000	0.000
9	Szerszám 9	0.0000	0.0000	0.0000	0.000
10	Szerszám 10	0.0000	0.0000	0.0000	0.000

18.3 G43 / G49 — How to use in G-code?

The G43 command "switches on" a tool's offset. From then on every motion is shifted automatically. The G49 command "switches it off" — back to the original state.

Parancs	What does it do?
G43 H1	Switches on the T1 tool offset (row 1 of the table)
G43 H2	Switches on the T2 tool offset
G43	Without H → uses the offset of the current T number
G49	Offset off — all shifts are zeroed

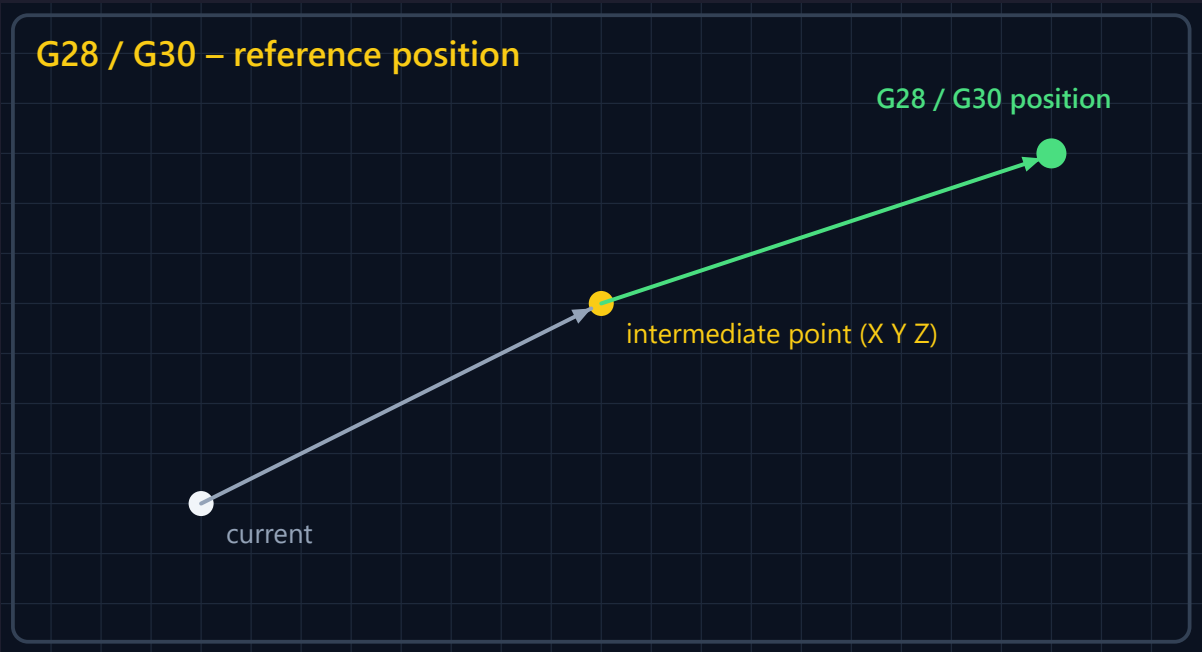
18.4 Full example — step by step

Suppose: T1 Z offset = -52.345 in the table. What does the machine do?

```
T1 M6 ; 🔧 Load tool 1
G43 H1 ; 📏 Switch on T1 offset
G0 Z10 ; ⬆ Programmed: Z10
; Real machine position: Z10 + (-52.345) = Z-42.345
G1 Z0 F200 ; ⬇ Programmed: Z0 → Real: Z-52.345
G49 ; ✖ Offset off – from now on Z0 = Z0
```

🔗 Bottom line: You only write "Z10", and the machine knows by itself where it really has to go!

19. G28/G30 — Reference position (Return home)



19.1 What is it and why is it needed?

The CNC machine has a "home" — a preset point you can send it back to any time. Typically this is the **top of the machine** (Z up), where you can change tools safely. G30 is a **second home** — for example the tool change position, if that is somewhere else.

🏠 Think of it as: "G28 = Go home!" / "G30 = Go to the garage!"

19.2 Setup

⚙ Settings → set the coordinates at the bottom of the Inputs & Home panel:

Position	X	Y	Z	What is it for?
----------	---	---	---	-----------------

G28	0	0	50	🏠 Machine reference (Z up)
G30	100	0	80	🔧 Tool change position

19.3 Commands in detail

Parancs	What does it do?
G28	Jumps straight to the G28 position (rapid speed)
G28 Z0	FIRST goes to Z0 → THEN to the G28 position. Safe, because Z is raised first!
G28 X5 Y5	First to X5 Y5, then to the G28 position (intermediate point)
G30	Jumps to the secondary reference position (G30)
G30 Z0	Through an intermediate point to the G30 position

19.4 Full example — tool change routine

After machining we change tool and continue:

```

; === Tool 1: Mill Ø6 ===
T1 M6 ; 🔧 Load T1
G43 H1 ; 📏 T1 offset on
G0 X10 Y10 ; Move into position
G1 Z-5 F200 ; Machining...
G0 Z10 ; Raise Z

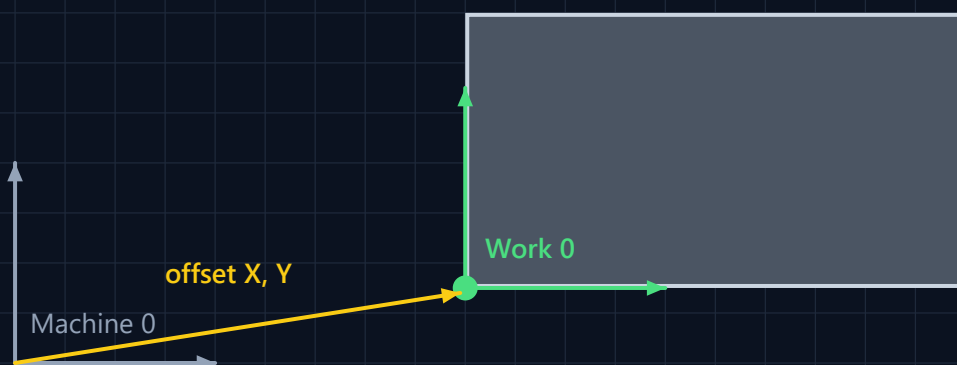
; === Tool change ===
G28 Z0 ; 🏠 Z home (intermediate: Z0, then G28 position)
T2 M6 ; 🔧 Load T2
G43 H2 ; 📏 T2 offset on

; === Tool 2: Drill Ø5 ===
G0 X20 Y20 ; To the new position
G1 Z-15 F100 ; Drilling...
G49 ; ✖ Offset off
G28 ; 🏠 Home
M30 ; ✅ Program end

```

20. G10 L2 — Setting the workpiece offset from G-code

Machine zero → work zero (G54 / G92 / G10 L2)



20.1 What is it and why is it needed?

Normally you set the workpiece zero (G54, G55, etc.) at the machine, by hand. But sometimes you want to say it **from the G-code program** : "The G54 X zero should be at 10mm." That is what the G10 L2 command is for — you set the offset from the program, no need to touch it manually!

💡 Imagine: "You write on a piece of paper that the zero point is here — and put it next to the machine. G10 L2 does the same, only from G-code!"

20.2 Syntax

```
G10 L2 P<number> [X..] [Y..] [Z..]
```

Ahol:

Parameter	Value	Meaning
L2	mandatory	Always L2 — this indicates workpiece offset setting
P1	→ G54	Workpiece coordinate system 1 (most common)
P2	→ G55	Coordinate system 2
P3	→ G56	Coordinate system 3
P4	→ G57	Coordinate system 4
P5	→ G58	Coordinate system 5
X/Y/Z	optional	Only the given axes are modified! The others stay unchanged.

20.3 Examples step by step

```
; Example 1: full G54 zeroing
G10 L2 P1 X0 Y0 Z0 ; all G54 axes = 0

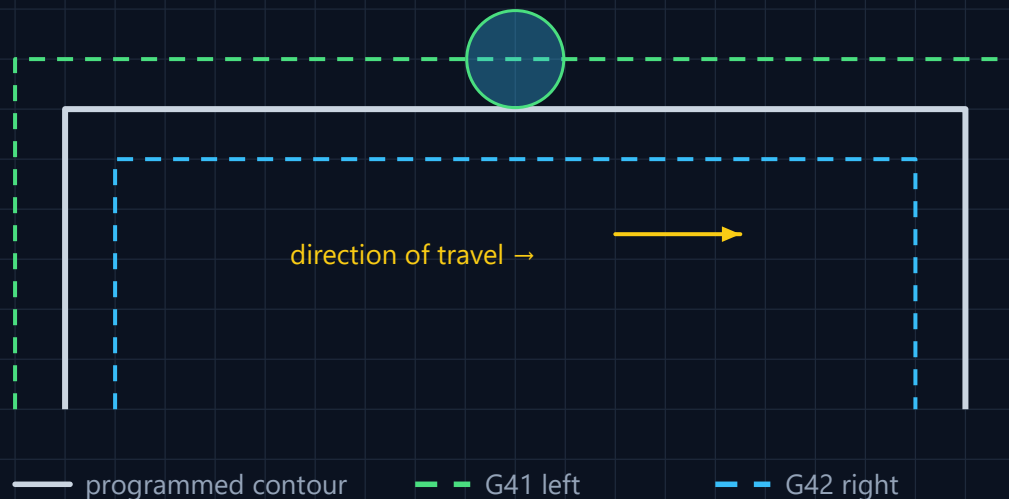
; Example 2: modify only the G54 Z offset
G10 L2 P1 Z-52.3 ; G54 Z = -52.3 (X and Y unchanged!)

; Example 3: several workpieces side by side
G10 L2 P1 X0 Y0 ; G54 = workpiece 1 (left)
G10 L2 P2 X100 Y0 ; G55 = workpiece 2 (100mm to the right)
G10 L2 P3 X200 Y0 ; G56 = workpiece 3 (200mm to the right)
;
G54 ; Switch to part 1
G0 X0 Y0 ; This is the X0 Y0 position of part 1
; ...machining...
G55 ; Switch to part 2
G0 X0 Y0 ; This is X0 Y0 of part 2 (actually X100!)
; ...the same machining...
```

💡 Tip: Works from MDI too! Type: G10 L2 P1 X0 Y0 Z0 → and the DRO updates immediately!

21. G40/G41/G42 — Cutter Radius Compensation

G41 / G42 – cutter radius compensation



21.1 What is it and why is it needed?

If you draw a square on paper and want to trace it with a **thick marker**, the pen does not go on the line but **next to it** — because the pen tip is thick! The same is true for the milling machine: the cutter is not a point, it has a radius (e.g. Ø6 = 3mm radius).

💡 Radius compensation means: "You draw the final shape, and the machine computes by itself where the cutter has to go next to it!"

Top view (XY plane):

G41 (left compensation): G42 (right compensation):

Programmed path: —→ Programmed path: —→
Cutter (left side): —→ Cutter (right side): —→
↑ r (radius) ↓ r (radius)

21.2 Commands

Parancs	What does it do?
G41 D3	Left compensation, 3mm radius (climb milling — the cutter runs on the LEFT side of the path)
G42 D3	Right compensation, 3mm radius (conventional — the cutter runs on the RIGHT side of the path)
G40	Compensation off — the cutter runs exactly on the programmed path

21.3 Where does the radius come from? (D parameter)

The D parameter can be interpreted in three ways:

What you write	Where does the radius come from?
G41 D3	1 Looks up the "Radius" field of T3 in the 🔗 Tool table. If it is > 0, uses it.
G41 D3	2 If the table Radius = 0, the D value is taken directly as the radius (3mm).
G41	3 Without D → looks up the table by the current T number.

21.4 Full example — milling a square

With a Ø6mm cutter (radius = 3mm) we mill a 50×50mm square:

```
T1 M6 ; 🔗 Mill Ø6
```

```

G43 H1 ; 📏 Tool offset
G0 X-10 Y0 ; Start left of the square
G1 Z-3 F200 ; Go down to depth
G41 D3 ; ⚙️ Left compensation ON (3mm radius)
G1 X0 Y0 F300 ; Corner of the square – cutter runs 3mm to the left
G1 X50 Y0 ; Bottom edge
G1 X50 Y50 ; Right edge
G1 X0 Y50 ; Top edge
G1 X0 Y0 ; Back to start
G40 ; ❌ Compensation off
G0 Z10 ; Raise Z

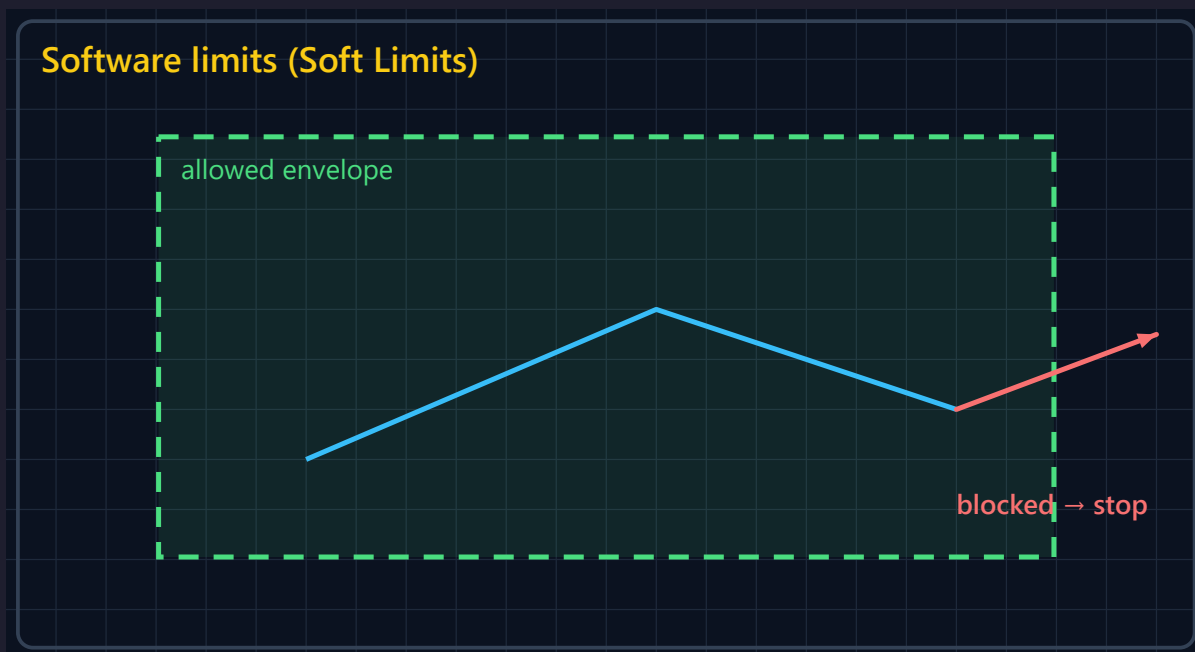
```

🔗 Result: You drew a 50×50mm square in the program, and the finished part will ALSO be exactly 50×50mm, because the machine automatically went 3mm aside!

21.5 Important rules

- **Works only in Mill mode** — has no effect in lathe mode
- **Compensates only in the XY plane** — has no effect on the Z direction
- **G2/G3 arcs are also compensated** — the arc radius grows/shrinks automatically
- **G0 rapid is also compensated** when G41/G42 is active
- **Always issue G40** at the end of the contour! Otherwise the next motion is compensated too.

22. Software limits (Soft Limits)



22.1 What is it and why is it needed?

Your CNC machine has a physical working area — e.g. the table is 500×500mm, the Z axis 100mm. If the program accidentally wants to go too far, **it can cause a collision** — the tool may break, the guide may slip off, or the machine may hit its end!

A **software limit** is an invisible "fence" around the machine. If the machine wanted to leave this area, the software **automatically stops it** at the boundary.

🛡️ Imagine: "There is an invisible rubber wall at the edge of the machine — it does not let it out!"

22.2 Setup

⚙️ Settings → 🛡️ Software limits panel:

Tengely	Min	Max	Meaning
X	-500	500	X axis travel range (mm)
Y	-500	500	Y axis travel range (mm)
Z	-100	100	Z axis travel range (mm)

☒ **Soft Limits enabled** checkbox — if not checked, the protection is off.

22.3 Where does it protect?

Helyzet	Behaviour	Example
 Jog stepping	Target position clipped	X=499, step=5 → X=500 (not 504!)
 Continuous jog	Stops at the boundary	Holding X+ → stops at X=500
▶ Program run	Segments clipped	G1 X600 → stops at X500
 MDI command	Also clipped	G0 Z-200 → stops at Z-100

 The log window always shows: "Soft limit warning: 3 lines clipped" — so you know if something would have gone too far!

22.4 Backlash compensation

When an axis changes direction (e.g. went right → now left), due to the backlash of the lead screw the first few steps spin "idle" — the axis does not move. The **backlash compensation** inserts extra steps on direction change to take out this play.

 Imagine: "The screw is a bit loose, so when you reverse it, it first spins a little before it grips. The compensation removes this spinning."

 Setting: Settings → Axes table → **Backlash** column (mm). Typical values: 0.01 – 0.1 mm. Zero = off.

Tengely	Backlash	Mit jelent?
X	0.05	On X direction change 0.05mm extra steps (10 steps at 200 step/mm)
Y	0.02	On Y direction change 0.02mm extra
Z	0.00	Z gets no compensation (ball screw, no backlash)

23. Subroutines

23.1 What is it and why is it needed?

Imagine you want to drill the same hole 100 times, only at different places. Instead of typing the same code 100 times, you write it once into a "recipe" (subroutine), and then you just tell the machine "do another one".

 Like a recipe in the kitchen: write it once → use it any time!

23.2 O-word syntax (recommended)

Parancs	What does it do?
O100 SUB	 Subroutine START — "100" is a name/number you choose
O100 ENDSUB	 Subroutine END
O100 CALL	▶ Run subroutine 100 (once)
O100 CALL L5	▶ Run it 5 times in a row!
O100 REPEAT L5	▶ Same as CALL L5 (alternative syntax)

23.3 M98/M99 syntax (compatibility)

Parancs	What does it do?
M98 P100 L3	Call subroutine 100 3 times (P = O-number, L = repeat)
M99	Return from subroutine (ENDSUB alternative)

23.4 Full example — drawing a square 4 times

We define a "draw a square" subroutine, then call it 4 times:

```
; === The "recipe" (subroutine definition) ===
O100 SUB ; 📄 Recipe start: "square 100"
G91 ; Relative mode (always relative to the current place)
G1 X10 F200 ; → 10mm right
G1 Y10 ; ↑ 10mm up
G1 X-10 ; ← 10mm left
G1 Y-10 ; ↓ 10mm down (back to start)
G90 ; Back to absolute mode
O100 ENDSUB ; 📄 Recipe end

; === Usage ===
G0 X0 Y0 ; Go to the start
G1 Z-2 F100 ; Go down to depth
O100 CALL L4 ; ▶ Draw 4 squares!
G0 Z10 ; Raise Z
```

🔗 Result: The machine draws 4 squares, but the code contains the square motions only once!

23.5 How does it work inside?

- The system **expands before running** the subroutines — inserts the code where you called it
- A **toolpath view** also shows the expanded result (you can verify visually!)
- The log indicates: "Subroutines expanded: 24 lines → 12 segments"

23.6 Limitations

Rule	Details
Nesting	Max. 10 levels (a subroutine may call a subroutine, which may call another, etc.)
Max lines after expansion	100 000 lines — infinite loop protection
Definition location	At the start OR end of the program — does not have to be before the call

23.7 External file subroutines

The subroutine can also be in a **separate file** — so frequently used routines (e.g. drill pattern, tool measurement) written once can be called from several programs. Two syntaxes are supported:

```
; — 1. Fanuc style: M98 (filename) —
M98 (subprogram_file.tap) L1 ; run external file once
M98 (drill_pattern) L3 ; also without extension – automatically tries: .tap .nc .ngc .gcode .txt

; — 2. Named O-word style —
O<external_file> CALL ; run the external file named "external_file"
O<external_file> CALL L5 ; repeated 5x
```

- **File lookup order:** 1) the folder of the loaded program, 2) %APPDATA%\CncController\Subprograms\ mappa
- Az **O<name> CALL** first looks for the **O<name> SUB ... O<name> ENDSUB** block defined in the program; if none, resolves it as an external file
- In the external file **M99** means return, the **M2/M30** does not stop the main program (it is left out of the callable body)
- The external file's own **O<name> SUB** definitions also become available for the calling program
- Works **from a G-code program, from MDI and from a macro** too (see 17.10)
- Protection: max. 10 nesting levels, self-calling file (loop) blocked, file cache for fast repeated calls

```
; Example from MDI:
M98 (frontplate_engrave.tap) L1

; Content of the external file (frontplate_engrave.tap):
```

```
G0 X0 Z2
G1 Z-0.5 F100
G1 X20
M99 ; return to the caller
```

24. Quick buttons, tool change mode, sounds

24.1 Quick buttons (Q1–Q10)

At the bottom of the main window there are 10 programmable **gyorsgomb** . Each can have a name and multi-line G-code — the code runs line by line on the full MDI channel (tool change M6 T., macros, external files also work in it). The lines are executed **sequentially** : the next line only starts when the previous motion/spindle operation has finished.

- **Editing:** right-click the button — enter name and code in the pop-up window
- **From keyboard:** Q lenyomva tartva + 1..9 = Q1..Q9, Q + 0 = Q10
- (Parenthesized) comment lines and % lines are excluded from execution
- While a quick button is running, another cannot be started (the log says: "Quick button: previous still running")
- E-Stop immediately aborts the remaining lines; after 5 minutes of waiting a timeout protection kicks in
- The buttons are stored in the %APPDATA%\CncController\quickbuttons.json file

24.2 Tool change mode (manual / automatic)

In Settings you can choose the **tool change mode: Manual** mode, at the WAIT commands of the M6 macro the machine stops and waits for Start (Alt+R) — you change the tool. **Automatikus** mode, the WAIT steps are skipped, the change runs without interruption (e.g. turret lathe or automatic tool changer).

24.3 Active tool display

Next to the DRO the active tool is shown (e.g. **T3**; if no tool: T—). The display is linked to the tool table: it shows the tool radius and description, and the G41/G42 radius compensation also takes its value from here if no D is given.

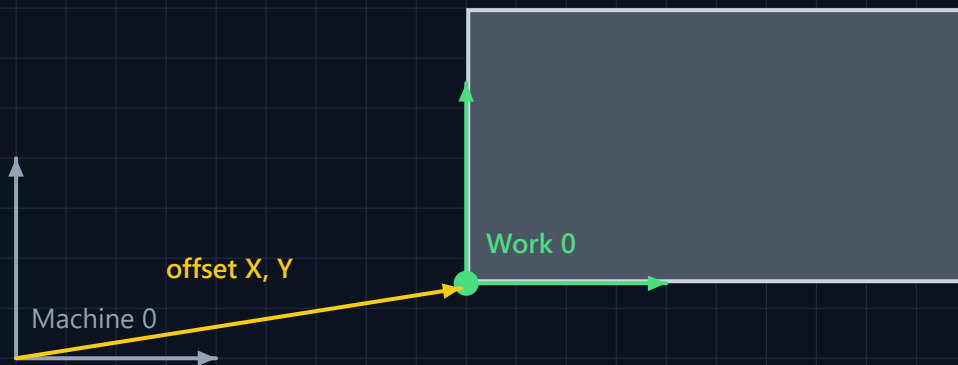
24.4 Sounds

The application signals important events with the built-in Windows system sounds — no separate sound files are needed:

- **Program start / Home start** — Asterisk sound
- **Pause / Resume** — Beep hang
- **Stop** — Exclamation sound
- **E-Stop activation / limit reached** — Hand (critical) sound
- **E-Stop release** — Asterisk sound

25. G92 — Coordinate offset

Machine zero → work zero (G54 / G92 / G10 L2)



25.1 What is it and why is it needed?

It happens that you do not clamp the workpiece exactly at the machine zero. Then the simplest way of zeroing is: **go to the desired point with the tool**, and tell the machine: "Let this be zero!" G92 does exactly that — from G-code!

💡 Imagine: "You take the ruler, put it on the corner of the workpiece and say: this is 0. It does not matter where the machine is — from here on this is the starting point!"

25.2 How does it work?

G92 does not move the machine — it only **renames** the current position. If the machine is currently at X=73.5 Y=42.1 Z=-10.0 and you issue:

```
G92 X0 Y0 Z0
```

From then on the DRO (digital readout) shows X=0 Y=0 Z=0 — but the machine physically stayed in the same place! The offset is stored, and every further coordinate is relative to this new zero.

🔗 Bottom line: G92 does not move — it only "relabels" the position!

25.3 Commands

Parancs	What does it do?
G92 X0 Y0 Z0	Let the current position be X=0, Y=0, Z=0 (full zeroing)
G92 X0 Y0	Zero only X and Y — Z stays unchanged!
G92 Z0	Zero only Z — X and Y unchanged
G92 X10 Y20	Let the current position be X=10, Y=20 (you do not necessarily set it to zero!)
G92.1	Clear the G92 offset — the original coordinate system is restored

25.4 Full example — step by step

The machine is currently at X=73.5 Y=42.1 Z=-10.0. We zero and work:

```
; === Zeroing at the workpiece corner ===
G92 X0 Y0 Z0 ; 💡 "This is zero!" – DRO: X0 Y0 Z0
; (the machine did NOT move, only the display changed!)

; === Machining relative to the new zero ===
G0 X10 Y10 ; This is the new X10 Y10 (actually X83.5 Y52.1)
```

```

G1 Z-5 F200 ; Plunge to 5mm
G1 X50 Y10 F300 ; Cutting...
G0 Z10 ; Raise Z

; === Restore ===

G92.1 ; ✗ Clear offset – DRO shows the original coordinates again
M30 ; ✔ Program end

```

25.5 Second example — Z zeroing only (lathe)

In lathe mode we touch the tool to the face of the workpiece and zero Z:

```

; Tool is at the face
G92 Z0 ; 📏 Z zero = face (X unchanged!)
G0 Z2 ; Z retract
G1 Z-30 F0.15 ; Turning 30mm from the face

```

25.6 G92 vs G10 L2 — what is the difference?

Szempont	G92	G10 L2
What does it do?	"Let the current position be this"	"Let the G54 zero point be this"
Relative/absolute	Computes the offset relative to the current position	Sets an absolute offset value
Which system?	Separate G92 offset (affects all coord. systems)	Modifies one of G54–G58 (with the P parameter)
Clearing	G92.1 — restores the original	G10 L2 P1 X0 Y0 Z0 — zero it by hand
When to use?	Quick zeroing at the machine	Multiple workpieces, precomputed offsets

💡 Rule of thumb: If you stand at the machine and want to zero quickly → G92. If you plan the zero points ahead in the program → G10 L2.

25.7 Important rules

- **Does not move!** — G92 never starts a motion, it only shifts the coordinate system.
- **Only the given axes** are modified — if you write only Z, X and Y stay unchanged.
- **You can give non-zero values too!** — G92 X10 means: "let the current position be 10" (not necessarily 0).
- **G92.1 clears everything** — restores the original coordinates (offset = 0).
- **Lathe diameter mode** — in G7 (diameter) mode the G92 X value is also a diameter, the system converts it to radius automatically.
- **Works from MDI too!** — Type into the MDI box: G92 X0 Y0 Z0 → the DRO goes to zero immediately.

⚠ Caution: If you forget G92.1 at the end of the program, the offset stays active on the next run! It is recommended to start with a clean slate using G92.1 at the beginning of the program.

26. Probing (G38.2 / G38.3)

G38.2 / G38.3 – probing



26.1 What is it and why is it needed?

Probing is the CNC machine's "sense of touch". With a sensitive probe the machine **stops on touch** and remembers where it found the workpiece or the tool tip. So you do not need to set the zero point by hand — the machine measures it for itself!

💡 Imagine: "The machine stretches out its finger, touches the workpiece and says: Got it, here is the surface!"

26.2 How does it work?

With the G38.2 / G38.3 command the machine **starts slowly** in the given direction and continuously watches the probe input on the LPT Status port. As soon as the probe signals a touch, the machine **stops immediately** and stores the touch position (ProbeX, ProbeY, ProbeZ).

The difference between the two commands:

- **G38.2** — If the probe does NOT touch anything along the whole way → **ALARM! The program stops.**
- **G38.3** — If it does not touch → **no problem, it continues** (silent measurement).

26.3 Commands

Parancs	What does it do?
G38.2 Z-50 F50	Probe downward in Z, max 50mm, F50 mm/min — ALARM if no touch
G38.3 Z-50 F50	Same, but does NOT alarm if no touch
G38.2 X10 Y10 F100	Probe motion in XY direction (side measurement)
G38.2 Z-30 F30	Typical tool length measurement: slowly down onto the setter block

26.4 Probe Status port bit configuration

The probe signal is read on the LPT Status port. It must be set in the **Settings** window:

Setting	Description
ProbeBit	Status port bit index (3–7). -1 = no probe connected
InvertProbe	Inversion for NC (normally closed) probe — true = NC, false = NO
ProbeMaxTravel	Maximum probe travel in mm (safety limit, 0 = no limit)

LPT Status port bits: S3 (Pin 15), S4 (Pin 13), S5 (Pin 12), S6 (Pin 10), S7 (Pin 11, HW inverted).

💡 Typical wiring: Probe signal → Pin 15 (S3 bit), ProbeBit = 3. For an NC probe InvertProbe = true.

26.5 Example — Z zeroing with a probe

We position the tool above the workpiece, then the probe goes down in Z and stops on touch:

```
; === Simple Z zeroing with probe ===
G90 ; Absolute mode
G38.2 Z-50 F50 ; ▾ Probe down, max 50mm, F50 mm/min
; (stops as soon as it touches the surface!)
G92 Z0 ; 💡 "This is Z zero!" — the probe touch point becomes Z=0
G0 Z5 ; ↑ Rise to 5mm
M30 ; ✅ Done — Z zero point set!
```

🔗 Result: The Z zero point is now exactly on the workpiece surface, measured by touch!

26.6 Tool length measurement automation

The most common use: **automatic tool length measurement**. There is a setter block (tool setter) at a fixed place, and at every tool change the machine goes down onto it, measures the tool length and writes the offset. No more manual measuring!

```
; === Automatic tool length measurement (from the M6 macro) ===
; Precondition: setter block at X100 Y50, Z0 is the top of the block
G0 Z50 ; ↑ Safe height
G0 X100 Y50 ; ➡ Above the setter block
G38.2 Z-80 F50 ; ▾ Probe down onto the block
; → The machine remembered: ProbeZ = the touch location
G92 Z0 ; 💡 Touch point = Z zero (or compute the offset)
G0 Z50 ; ↑ Back to safe height
```

💡 Pro tip: You can build this measurement into the M6 (tool change) macro, so it runs automatically after every change!

26.7 Probing from a macro (PROBE command)

The macro language also supports probing with the PROBE parancssal:

```
PROBE Z-50 F50 ; Probe motion in Z direction
```

After the measurement the probe variables are available:

Variable	Meaning
#probe_tripped	1 = there was a touch, 0 = none (the probe went all the way)
#probe_x	X coordinate of the touch point
#probe_y	Y coordinate of the touch point
#probe_z	Z coordinate of the touch point

26.8 Important rules

- **The probe input must be configured!** — with ProbeBit = -1 G38 does not work.
- **F (feed rate) is mandatory!** — The probe always moves slowly, at the given feed. Typical: F30–F100.
- **G38.2 raises an alarm** if the probe does not touch — this is intentional, because something is wrong!
- **G38.3 continues silently** — use it if a touch is not certain (e.g. position search).
- **After probe touch the machine STOPS** — the position is stored, but the machine does not retract automatically. You have to give the retract (G0 Z...).
- **EStop check is active** during probing too — safety stop works any time.
- **NC (Normally Closed) probe** — For industrial probes set InvertProbe = true so that the level change means touch.

⚠ Caution: Always probe slowly (F50 or less)! At fast motion the probe may break, or stopping accuracy degrades due to inertia!

27. Run from line N

27.1 What is it and why is it needed?

It happens that the machine stops in the middle of the program (breakage, power failure, error) — then you do not want to run the whole program from the start. The **Run from line** function lets you **start execution from any line** , while the machine correctly restores the state (spindle, coolant, mode, offsets).

💡 Imagine: "You press pause in the middle of a movie. When you continue, you do not have to watch it from the start — but you need to know the context of the story. The machine does exactly this: it recomputes what happened up to the given line."

27.2 How does it work inside?

When you select the line and start, the system works **in 4 steps** dolgozik:

1. **State simulation** — From line 1 of the program to the selected line it "runs the G-code in its head". It does not move the machine, it only computes: what would the spindle state be? Which mode is active (G90/G91)? What feed rate? What offsets? Which coordinate system?
2. **Position check** — Compares the simulated position with the machine's current position. If there is a difference (e.g. you moved the machine by hand), it writes a warning to the log.
3. **State application** — Turns on the spindle, coolant, sets the coordinate system — exactly as it would be in the middle of the program.
4. **Execution** — Plans and executes the motions from the selected line to the end of the program.

27.3 How to use it?

1. Click in the G-code editor on the line you want to start from.
2. Right-click → "▶ **Run from here (Run from line)**"
3. The system simulates the state, turns on spindle/coolant if needed, and starts!

💡 The cursor (caret) position determines the line number — no need to select, it is enough that the cursor is in the desired line!

27.4 What does the system simulate?

The simulation recomputes the following states:

State	What happens?
Position (X, Y, Z)	Computes where the machine would be — compares with the current position
Spindle (M3/M4/M5)	If it should be running → turns it on, if not → turns it off
Coolant (M7/M8/M9)	Turns it on if active according to the simulated state
Mode (G90/G91)	Restores absolute or relative mode
Feed (F)	The last set feed rate becomes valid
Coordinate system	G54–G58 active system, work offsets, G92 offset
Tool compensation	G43 tool length, G41/G42 cutter comp — continues if it was active
Path mode	G61 (exact stop) or G64 (continuous) and tolerance

27.5 Example

The machine stopped at line 20 of the program. You changed the tool, restored the position and continue from line 20:

```
; Program content (excerpt):  
N10 G90 G54 ; Absolute mode, G54 system
```

```

N11 M3 S12000 ; Spindle on 12000 RPM
N12 M8 ; Coolant on
N13 G0 X0 Y0 Z5 ; Start
... ; (lines 14-19)
N20 G1 X50 Y30 F500 ; ← You want to continue HERE
N21 G1 X50 Y60 ; ...

```

Right-click line 20 → "Run from here" — the system:

- Simulates lines 1–19 → Knows: G90, G54, M3 S12000, M8, F500
- Turns on the spindle at S12000 RPM and the coolant
- Checks the position (warns if different)
- Starts execution from line 20!

27.6 Log messages

When starting the run you can see detailed information in the log window:

```

▶ Run from line 20 – Simulation: 19 lines processed
Simulated position: X=45.000 Y=25.000 Z=-3.000
Mode: G90, Coord: G54, F500
Spindle: M3 S12000
⚠ WARNING: The machine position differs from the simulated one (12.500 mm)!
Program started (from line 20)

```

27.7 Important rules

- **The machine does NOT move to the starting position!** — It starts from the current position, not the simulated one. Set the machine beforehand by hand (jog) or MDI!
- **Position warning** — Ha a gép pozíciója eltér a szimulálttól (>0.1mm), the log indicates it. This is normal, but be aware!
- **Spindle/Coolant automatic** — If according to the simulation the spindle should be running, it turns on automatically at the proper speed.
- **Subroutine expansion** — If the selected line is in the middle of a subroutine, the system works from the expanded lines.
- **The context menu is available in the editor** — Not while running, but after a stop: right-click the desired line!

⚠ Caution: If there was a tool change (M6) before the selected line and the tool is different, you have to change it by hand! The simulation sets the tool offset, but does not physically change the tool.

28. Single Step + Block Delete

28.1 What is it and why is it needed?

Two complementary functions that are essential for **program testing and debugging** :

- **Single Step (stepping mode)** — The program runs line by line: executes one line, stops, and waits for the "Start" button. So you can see what the machine does in every single line.
- **Block delete (/)** — Lines starting with a slash (/) are skipped by the program. So you can put optional steps into the G-code that you can enable or disable as needed.

💡 Imagine: Single Step is like a frame-by-frame video — you can watch it frame by frame. Block Delete is like the sentences in brackets in a book — you only read them if you want to.

28.2 Single Step (stepping mode) — line-by-line execution

When stepping mode is active, the program **stops after every G-code line** and waits. Pressing the "Start" button steps to the next line.

How to activate?

1. Click the **1** button on the top button row (turns yellow when active)
2. Press the **► Start** button — the program starts and stops after the first line
3. Every "Start" press = execution of one line
4. If you want to turn it off while running: click the **1** button again → the program continues continuously

In the log window you see which line it is standing at:

```
► Single Step: line 5 done – waiting
► Single Step: line 6 done – waiting
```

💡 Tip: Stepping mode can be toggled while running too! If you are tired of stepping, turn it off and the program runs through.

28.3 Block Delete (/) — skipping slash lines

If a G-code line **starts with a / (slash) character**, it is an "optional line":

- **Block Delete OFF** — The slash lines execute normally (the / sign has no effect)
- **Block Delete ON** — The slash lines are **COMPLETELY skipped**, as if they were not there

Example G-code:

```
G90 G54
M3 S12000
/ M8 ; ← Optional coolant (only if Block Delete OFF)
G0 X0 Y0 Z5
G1 Z-3 F200
/ G1 X50 Y50 F100 ; ← Optional extra pass
G0 Z50
M30
```

If Block Delete ON → the "/ M8" and "/ G1 X50 Y50 F100" lines are skipped. If Block Delete OFF → every line runs.

How to activate?

1. Click the **/BD** button on the top button row (turns purple when active)
2. Start the program — the / lines are skipped automatically
3. The log indicates: "Block Delete: N lines skipped (/)"

28.4 Typical use cases

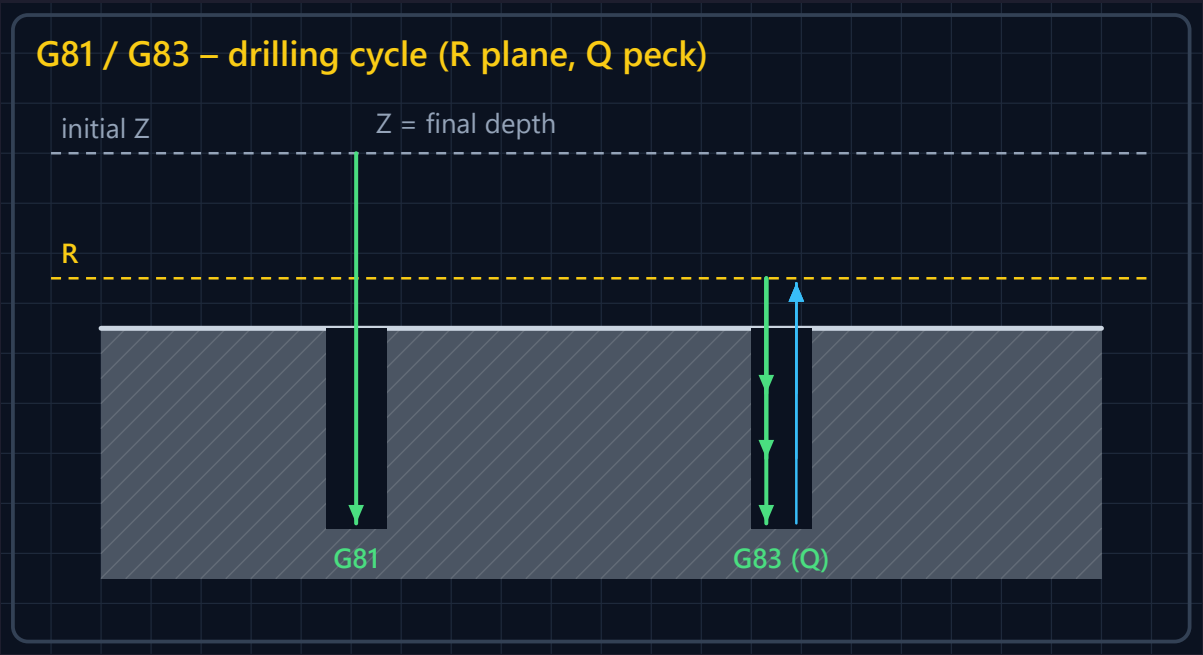
Eset	Which to use?
Testing a new program	Single Step — you watch line by line what happens
Optional coolant on/off	Block Delete — / M8 line in the program
Skipping an extra finishing pass	Block Delete — / G1... lines for the finisher
Examining machine behaviour	Single Step — you see what it does in every line
Debugging after a crash	Both + Run from line (chapter 27)

28.5 Important rules

- **Single Step can be toggled while running!** — Turn it on, step, turn it off → the program continues normally.
- **Block Delete must be turned on BEFORE STARTING the program!** — The filtering happens in the planning phase, the / lines are left out of the segment plan too.
- **The slash (/) must be at the very start of the line!** — Space or other character before it → does not count as a block delete line.
- **Block Delete + Single Step work together too!** — The / lines are skipped, the rest runs line by line.
- **The Start button is the stepper** — In Single Step mode the Start button = "next line".

⚠ Caution: In Single Step mode the spindle keeps running and the coolant stays active between lines too! If you want to stop it, press the Stop button!

29. Canned Cycles — Drilling cycles (G81–G89)



29.1 What is it and why is it needed?

Canned cycles are **automated drilling, tapping and boring** motion sequences. A single G-code line performs several motions: XY positioning, Z plunge, retract — all automatically.

💡 Imagine: Instead of writing 5-6 lines for every hole (rapid there, rapid down to R, feed down to Z, rapid back), you solve it with a single line: "G81 X10 Y20 Z-15 R2 F200" — and the machine does everything!

29.2 General operation — Every cycle works in 4 steps

1. **Rapid XY** — Rapid move to the hole position (X, Y)
2. **Rapid Z → R plane** — Rapid move down to the R (reference/retract) plane
3. **Feed Z → depth** — Feed move to the cycle-specific Z depth (the cycle type determines the details)
4. **Retract** — Retract to the R plane (G99) or to the starting Z (G98)

29.3 Parameters

Parameter	Meaning
X, Y	Hole position (the machine goes there with rapid)
Z	Final depth of the hole (the cycle drills down to here)
R	R plane height (reference / retract plane, e.g. R2 = above Z2)
F	Feed mm/min (drilling speed)
Q	Peck depth (G83 peck drill — e.g. Q3 = retract every 3mm)
P	Dwell at the hole bottom in seconds (G82, G88, G89)

29.4 Supported cycles

G81 — Simple drilling

Rapid XY → Rapid R → Feed to Z depth → **Rapid retract**. The simplest drilling cycle.

```
G81 X10 Y20 Z-15 R2 F200 ; Hole: X10 Y20, depth Z-15, R plane Z2
```

G82 — Drilling with dwell

Like G81, but at the hole bottom **waits P seconds** (chip removal, surface improvement).

```
G82 X10 Y20 Z-15 R2 F200 P0.5 ; + 0.5s dwell at the hole bottom
```

G83 — Peck drilling

Drills down by Q depth, retracts to the R plane (chip removal), then plunges again — repeats until it reaches the Z depth. Essential for deep holes!

```
G83 X10 Y20 Z-30 R2 Q5 F150 ; 5mm pecks, 30mm deep hole
```

💡 Q5 means: drill 5mm, retract to R, rapid back to 1mm above the previous depth, drill another 5mm... and so on until Z-30 is reached.

G84 — Tapping

Feed down to Z depth → **Spindle reversal** → feed up to the R plane → **spindle reverses back**. The feed (F) and speed (S) follow from the thread: $F = S \times \text{thread pitch}$.

```
G84 X10 Y20 Z-12 R2 F500 ; Tapping M6x1 (S500, F=500mm/min)
```

⚠️ For tapping the spindle must already be running (M3)! The cycle reverses it and turns it back automatically.

G85 — Boring — Feed retract

Feed down to Z → **feed retract** (not rapid!). Smoother surface than G81, because the retract is a feed move too.

```
G85 X10 Y20 Z-15 R2 F100 ; Boring, feed retract
```

G86 — Boring — Spindle stop, rapid retract

Feed down to Z → **spindle stops** → rapid retract → **spindle restarts**. Prevents scratching the hole surface during retract.

```
G86 X10 Y20 Z-15 R2 F100 ; Bore, spindle stop + rapid retract
```

G87 — Back boring

Special cycle for back boring. Simplified implementation: feed down + rapid retract.

G88 — Boring + dwell + manual retract

Feed down to Z → **P dwell** → **manual stop** (M1-like pause — continue with Start button) → retract. The operator can inspect the hole before retracting.

G89 — Boring + dwell + feed retract

Feed down to Z → **P dwell** → **feed retract** (not rapid). Combines the advantages of G82 and G85.

```
G89 X10 Y20 Z-15 R2 F100 P1 ; Bore + 1s dwell + feed retract
```

29.5 G80 — Cycle cancel

A **G80** command cancels the active drilling cycle. After that XY motions do not repeat the cycle.

```
G80 ; Cycle cancel — from here on no automatic drilling
```

29.6 G98 / G99 — Retract mode

Code	Retract target
G98	Retract to the starting Z (the position before the cycle) — safely high
G99	Retract to the R plane (faster, minimal motion between holes) — default

💡 Use G98 if there are obstacles between the holes (clamps, steps). G99 is faster if the surface is flat.

29.7 Sticky parameters and repetition

The canned cycle parameters are "**sticky**": you give them once, and on the following lines only the new XY coordinates are needed — the rest (Z, R, F, Q, P) is kept!

```
G83 X10 Y10 Z-30 R2 Q5 F150 ; Hole 1: full parameters
X30 Y10 ; Hole 2: only XY changes
X50 Y10 ; Hole 3: again only XY
X50 Y30 Z-20 Q3 ; Hole 4: XY + new depth + smaller peck
G80 ; Cycle end
```

29.8 Summary table

Ciklus	Name	Retract	Special
G81	Simple drilling	Rapid	—
G82	Drilling + dwell	Rapid	P dwell at the bottom
G83	Peck drilling	Rapid	Q peck depth
G84	Tapping	Feed (backwards)	Spindle reversal
G85	Boring	Feed	Smooth surface
G86	Bore + spindle stop	Rapid	Spindle stops → restarts
G87	Back bore	Rapid	Special
G88	Bore + dwell + manual	Rapid	P dwell + pause
G89	Bore + dwell + feed	Feed	P dwell + smooth retract

29.9 Full example — plate with 4 holes

```
; 4 holes in a plate: G83 peck drilling
G90 G54 ; Absolute, G54 system
M3 S3000 ; Spindle on
G0 Z20 ; Safe height
G99 ; Retract to R plane
G83 X20 Y20 Z-25 R2 Q5 F150 ; Hole 1
X60 Y20 ; Hole 2 (sticky params)
X60 Y60 ; Hole 3
X20 Y60 ; Hole 4
G80 ; Cycle end
G0 Z50 ; Retract
M5 ; Spindle off
M30 ; Program end
```

29.10 Important rules

- **G80 is mandatory at the end of the cycle!** — Without it every XY motion starts another drilling!
- **The R plane must be above the workpiece!** — The R value is the Z height where drilling starts. If R is inside the material → problem!
- **G83 Q value should be smaller than the drill diameter** — Recommended: $Q = \text{drill_diameter} \times 0.5\text{--}1.0$
- **For G84 tapping: $F = S \times \text{thread pitch}$!** — E.g. M6×1 thread at S500: $F = 500 \times 1 = 500$ mm/min
- **G98 vs G99: when in doubt, G98 is safer** — Retracts to the starting height, avoids the clamps
- **The parameters are "sticky"!** — Only give them again if you want to change them (Z, Q, F etc.)

⚠ Vigyázat: A fúróciklus NEM ellenőrzi a fúró törését! Mély furatoknál ($>3 \times \text{diameter}$) always use G83 pecking, and watch chip removal (coolant, air blast)!

30. G-code editor

30.1 Overview

The built-in G-code editor has **three main functions** that make writing, reading and debugging G-code easier:

- **Syntax highlighting** — Color coding: G-codes, M-codes, coordinates, parameters, comments each in a different color
- **Line numbering** — Continuous line numbers on the left side of the editor, always in sync with the content
- **Auto-complete** — Pop-up suggestion list while typing for G/M codes and parameters, with description

30.2 Syntax highlighting — Color codes

The editor automatically recognizes and colors the G-code elements. The highlighting refreshes with a 200ms delay after the last keystroke (does not slow down typing).

Element type	Color	Example
G-codes (G1, G90...)	 Blue	G1 G90 G54 G43
G0 Rapid move	 Yellow	G0
G2/G3 Circular move	 Cyan	G2 G3
M-codes (M3, M5...)	 Lila	M3 M5 M8 M30
Coordinates (X, Y, Z, I, J, K)	 Green	X10.5 Y-20 Z5 I0 J15
Parameters (F, S, R, Q, P, H, D, L)	 Pink	F200 S12000 R2 Q5 P0.5
Comments (; ...)	 Gray	; This is a comment
Line numbers (N...)	 Faint	N10 N20 N100
O-word (subroutines)	 Narancs	O100 SUB, O100 CALL
Block delete (/)	 Piros	/ M8

30.3 Line numbering

On the left side of the editor gray numbers show the line numbers. This is **not the same as the N numbers** in the G-code — this is the physical line number (1, 2, 3...), always in sync with the content.

- The line number refreshes automatically as you type (adding/deleting lines)
- The "Run from here" (Run from line) function uses these line numbers
- While running, the ListBox view shows the active line (highlighted)

30.4 Auto-complete — Typing assistance

When you type G, M, X, Y, Z, F, S or another parameter letter, a **pop-up suggestion list** appears with the possible codes and a short description.

Usage:

1. Start typing the code (e.g. "G8")
2. The suggestion list appears (G8 → G80, G81, G82, G83... etc.)
3. **↓/↑ arrow** — navigate in the list
4. **Enter / Tab** — insert the selected item
5. **Escape** — close the list
6. Keep typing → the list narrows automatically

Example — list appearing after typing "G8":

```
G80 – Canned cycle cancel
G81 – Drilling cycle
G82 – Drilling + dwell
G83 – Peck drilling
G84 – Tapping
```

...

💡 The autocomplete knows 70+ G-codes, M-codes and parameters, each with a description!

30.5 Key combinations in the editor

Key	Function
Enter / Tab	Insert autocomplete item (if the list is open)
↓ / ↑	Autocomplete list navigation
Escape	Close autocomplete / Leave the editor
Jobb klikk	Context menu (Run from here, Cut, Copy, Paste)

30.6 Technical details

- **Highlighting works with a 200ms delay** — During fast typing it does not refresh on every key, so there is no slow-down
- **The cursor position is preserved** — When the highlighting refreshes, the cursor returns to its original position
- **RichTextBox based** — Syntax highlighting is based on a WPF RichTextBox, which allows several colors in a single text field
- **Regex tokenizer** — Syntax analysis is done with regular expressions: every token type (G-code, M-code, coordinate, comment etc.) has its own rule

⚠️ Note: For very long programs (10,000+ lines) the syntax highlighting may cause some delay. In that case it is worth loading the G-code from file (Ctrl+O) — the highlighting runs once.

31. Program time estimation

31.1 Overview

On the left panel three time displays show the state of the machining:

Display	Color	Meaning
Expected	■ Green	Total estimated machining time (with acceleration, dwell)
Eltelt	■ Yellow	Real elapsed time (pause does not count)
Remaining	● Piros	Estimated remaining time based on the current segment

31.2 Taking acceleration/deceleration into account

The time estimation uses a **trapezoidal motion profile** , which gives a more realistic picture of the actual machining time:

- **Acceleration phase** — The machine accelerates from 0 to the target speed (F feed). Time: v / a
- **Constant phase** — Travels at constant speed. Time: $\text{remaining_distance} / v$
- **Deceleration phase** — The target speed decelerates to 0. Time: v / a

If the segment is too short to reach the target speed (triangular profile), the time estimation handles that too: $t = 2 \times v / (d / a)$.

💡 The acceleration value comes from the Settings → "Acceleration (mm/s²)" field. The higher the acceleration, the shorter the estimated time. Default: 100 mm/s².

31.3 What does the time estimation include?

Elem	Included?	Hogyan
G0 rapid move	✓ Igen	Rapid feed rate + acceleration
G1/G2/G3 feed move	✓ Igen	F value + acceleration
G4 Dwell (wait)	✓ Igen	P parameter (seconds)
G81–G89 drilling cycles	✓ Igen	Every segment individually

Feed/Rapid override	✗ No	The estimate assumes 100%
M1 optional stop	✗ No	Operator dependent, cannot be estimated
M6 tool change (macro WAIT)	✗ No	Operator dependent, cannot be estimated

31.4 Remaining time — How does it work?

The "Remaining" display refreshes while running (every 33ms):

1. During planning the **cumulative time** is computed for every segment (total time from the program start to that segment)
2. While running, the polling timer queries the **current segment index** from the StepGenerator
3. Remaining = Total estimated time – Cumulative time up to the current segment

💡 The remaining time is computed from the estimate, not from real time. If the override is at 50%, the actual time will be twice the estimate, but the remaining display does not know that — watch the real elapsed time next to it!

31.5 States

State	Remaining	Explanation
Not running	--:--:--	No active program
Start	= Expected	Starts with the total estimated time
While running	Decreasing	Refreshes per segment
Finish	00:00:00	Program ended successfully
Stop	--:--:--	Resets to default

⚠ The estimate is always optimistic: it does not count tool change time, operator waiting (M1), or feed override. Real time = estimated × (100 / override%).

32. GRBL USB controller support

Besides the LPT port the application also supports **GRBL 1.1** compatible USB CNC controllers (Arduino + GRBL firmware, CNC Shield etc.). In GRBL mode the G-code lines go directly to the controller by streaming — step generation, acceleration planning and motion control happen in the GRBL firmware.

32.1 Setup

1. Settings → Connection type → **GRBL (USB serial port)**
2. Select the COM port (e.g. COM3) and the baud rate (default: 115200)
3. Click Save — the application connects automatically

💡 With the  button you can refresh the list of available COM ports (after connecting the USB cable).

32.2 Operational differences (LPT vs GRBL)

Function	LPT mode	GRBL mode
Motion control	PC generates the step pulses	GRBL firmware (real-time)
Jog (manual motion)	StepGenerator + LPT	\$J command (jog cancel: 0x85)
Homing	Status port limit reading	\$H command (handled by GRBL)
Pause / Resume	StepGenerator.Pause/Resume	Feed Hold (!) / Cycle Start (~)
Emergency stop	E-Stop input + software stop	Soft Reset (0x18) + ALARM monitoring
Position query	Step counter (StepGenerator)	? status query (MPos/WPos)
Spindle control	LPT pin + optional PWM	M3/M5 G-code command

32.3 GRBL real-time commands

- ? — Status query (position, state, feed, override)

- **!** — Feed Hold (immediate stop, position kept)
- **~** — Cycle Start (continue after hold)
- **0x18** — Soft Reset (full stop, GRBL restart)
- **0x85** — Jog Cancel (immediate stop of jog motion)
- **\$H** — Homing cycle (GRBL firmware handles the limit switches)
- **\$X** — Alarm release (unlock)

32.4 Recommended GRBL settings

The \$-settings of the GRBL firmware can be modified directly through the MDI box (e.g. **\$110=2000** for the X axis max speed). The most important parameters:

- **\$100-\$102** — Steps/mm (X, Y, Z axis)
- **\$110-\$112** — Max speed mm/min (X, Y, Z)
- **\$120-\$122** — Acceleration mm/s² (X, Y, Z)
- **\$22=1** — Enable homing cycle
- **\$10=1** — Status report: MPos (machine coordinates)

33. STB4100 USB controller (RNR ECO MOTION)

The application supports the **STB4100** USB HID motion controller card (RNR ECO MOTION 2.0, RobotAndRobot.com). With the connection type in Settings set to **STB4100** the step generation is done by the card — the PC streams the planned segments over USB.

33.1 Setup

- Settings → Connection type → **STB4100**
- The device is detected automatically by its product name (VID_A720 / PID_F803)
- On connection the log shows: "STB4100 connected: ... SN=..."

33.2 Operational characteristics

- **Motion:** G0/G1/G2/G3 segments streamed to the card; jog, MDI, full program run supported
- **Homing:** own USB-based home sequence (not the LPT HomingSequence)
- **Bemenetek:** I1–I4 inputs readable (for limit, probe, E-Stop wiring)
- **Feed display:** the card does not report live feed — the displayed value is an estimate computed from the position change (smoothed)
- **Macros, tool change, external file subprograms** work the same as in LPT mode

Note: LPT-specific functions (e.g. PC-side step generation, encoder based G33) depend on the card's capabilities in STB4100 mode — unsupported functions are indicated in the log.

34. C axis (spindle position control)

When the spindle is configured in Step/Dir servo mode (Settings → Spindle drive type → Step/Dir servo), C axis position control is available automatically. The DRO panel shows the **C** position display in degrees.

34.1 Modes

The spindle switches automatically between two modes:

Mode	Description
Continuous rotation	On M3/M4 the spindle rotates continuously at the speed given by the S parameter Normal turning/milling.
C axis position	After M5, if a G0/G1 command contains a C parameter, the spindle switches to position control and moves to the given angle.

34.2 G-code usage

G-code	Function
M3 S1000	Spindle on → continuous rotation 1000 RPM
M5	Spindle stop (after this the C axis can be used)
G0 C0	C axis rapid positioning to 0°
G1 C90 F100	C axis controlled move to 90°
G1 X5 Z-2 C180	Simultaneous XZ motion + C rotation
M3 S800	Switch back to continuous rotation → C mode turns off automatically

34.3 Typical usage

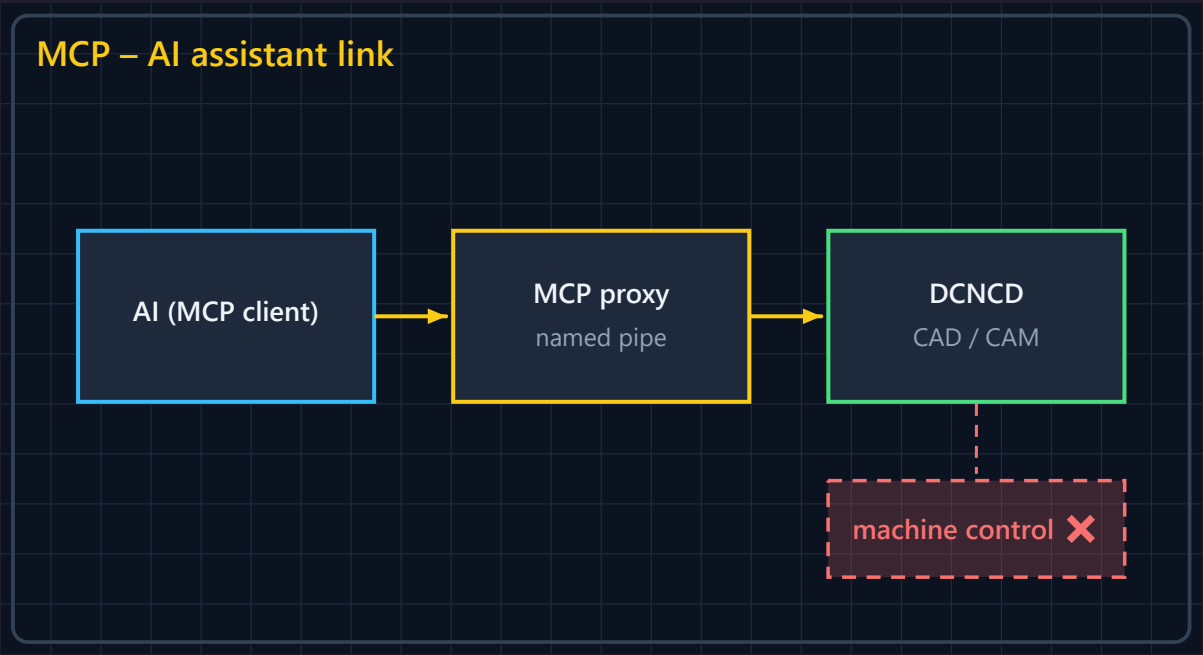
- **Milling on a lathe (C axis + Y/X)** — Milling slots, flats, hexagon profiles on the workpiece by indexing the spindle
- **Angle-positioned drilling** — Radial holes with equal angular spacing (e.g. 0°, 120°, 240°)
- **Engraving on a rotating part** — Text or pattern on the cylindrical surface by coordinating the C axis + Z

Note: The C axis DRO only appears if the spindle is configured in Step/Dir mode. The 0 button zeroes the C position. Absolute/incremental mode (G90/G91) applies to the C axis too.

35. Notes

- If the LPT port driver is not available, the application automatically enters simulation mode (see chapter 16).
- The settings are stored in the %APPDATA%\CncController\pinconfig.json file.
- The macros are stored in the %APPDATA%\CncController\Macros\macros.json file.
- In lathe mode the X axis means the radius; in diameter mode (G7) the display and the G-code X values are diameters.
- The arc commands (G2/G3) automatically use the XZ plane in lathe mode (I/K), and the XY plane in mill mode (I/J).
- The G33/G33.1/G33.2 commands need a spindle encoder (Settings → Encoder section). Minimum: Index signal; recommended: A/B + Index.
- The pause button (Space) stops the motion immediately even mid-segment, and resume starts from the same place.
- At an M1 optional stop the status bar shows the state, the Start button (or Alt+R) continues the program.
- In GRBL mode the G-code goes directly to the controller by streaming — the PC does no step generation.

36. MCP — AI assistant integration (CAD/CAM)



On startup the application opens a local **MCP (Model Context Protocol)** server through which AI assistants (GitHub Copilot, Claude, Gemini CLI, Codex CLI, etc.) can draw directly in the CAD editor, add CAM operations and generate G-code — from a natural-language request.

Security boundary: the MCP surface is CAD/CAM ONLY. Machine control (jog, program start, port, spindle, settings) is not reachable through it — such requests are rejected by the server. Generated G-code only goes to the editor; sending it to the machine remains a manual action with the Start button.

36.1 How it works

- The server runs on the `\\.\pipe\CncController.Mcp` named pipe; it starts and stops with the main window. No network port is opened.
- AI clients use stdio transport, so a small proxy (**CncController.McpProxy.exe**) forwards the messages to the pipe. The proxy interprets nothing and adds nothing.
- Every operation runs on the UI thread, is **undoable** (Ctrl+Z), and is visible immediately on the drawing / in the operation list.
- If you restart the application, restart the MCP server in the client too (the proxy exits when the pipe is closed).

36.2 Available tools

Tool	Description
<code>cad_list_entities</code>	All drawing entities with index, type, layer and selection state
<code>cad_add_entities</code>	Add point, line, rectangle, circle, arc, ellipse, polyline, text (layer can be given)
<code>cad_delete_entities</code>	Delete entities by index
<code>cad_select</code>	Select by index (the selection can serve as CAM operation geometry)
<code>cad_clear</code>	Clear the whole drawing (undoable)
<code>cam_list_operations</code>	The mill CAM operation list with all parameters
<code>cam_add_operation</code>	Add profile, pocket, drill, engrave, groove, V-Carve, 3D roughing/finishing operation with tool, feed, depth and strategy parameters
<code>cam_remove_operation</code>	Remove an operation by index
<code>cam_generate_gcode</code>	Generate G-code from the enabled operations — same as the Generate button: the code is loaded into the Toolpath view and the G-code editor, and is also returned as text to the AI

36.3 Extended CAD/CAM tools

- `cad_edit` — a single tool that works on the selected entities according to the `op` parameter. If `indices` is given, those are selected first. Every step is undoable and no dialog is opened. Supported:
 - move, copy, rotate, scale, mirror (also as copy);
 - offset, fillet and chamfer (two entities or all corners);
 - explode, join, reverse direction;
 - union, subtract, intersect on closed shapes;
 - rectangular and polar array;
 - move entities to a layer, delete duplicates, Vector Doctor, draw spiral;
 - undo and redo.
- `cad_list_layers` and `cad_layer` — list, add, delete layers and make a layer active.
- `cad_import_file` — import DXF, SVG, PDF and images; STL or other meshes go to the CAM panel.
- `cad_export_dxf` — save the selection, or the whole drawing if nothing is selected, to DXF.
- `cam_update_operation` — replace the parameters of an existing CAM operation. If no new geometry is given, the old one is kept.
- `cam_move_operation`, `cam_enable_operation`, `cam_clear_operations` — operation order (this defines the G-code order), enable/disable, and clearing the whole list.

36.4 Usage — example session

1. Start the DCNCD application and switch to the **Mill CAM** tab.
2. Start / enable the cnccontroller-cadcam server in the AI client.
3. Describe the task in natural language, for example:

Draw a 48x65 mm rectangle from the (0,0) corner and a Ø30 circle in its center. With the rectangle and the circle selected add a Pocket operation: T1 6 mm tool, S1200 F800, 2 mm depth per pass, 5 mm total depth, 40% stepover, then generate the G-code.

The assistant then calls `cad_add_entities` → `cad_select` → `cam_add_operation` → `cam_generate_gcode` in sequence. The drawing, the operation list and the Toolpath view refresh immediately; the circle is handled automatically as an island (chapter 13.3). Check and edit the result exactly as you would with manually added operations — AI-created items are no different.

36.5 Troubleshooting

- **“Failed to connect...”** — the application is not running. Start it, then restart the server in the client.
- **“MCP server process exited unexpectedly”** — the application was restarted and the proxy exited. Restart the server in the client.
- **The tool fails with “not allowed”** — the requested action would be machine control; this is intentionally blocked.
- **Access denied on the pipe** — the application runs elevated, the client does not; the pipe permissions handle this by default, but Windows policy may override it.