Twinwright Studio 1.2.1

Twinwright Studio manual

2026-10-16

Inhalt

User manual

1. Erste Schritte
2. Das Fenster und die Arbeitsbereiche
3. Maschinen und Einschalten
4. Einen Auftrag einrichten
5. Antasten und den Werkstücknullpunkt finden
6. Programme erstellen (CAM)
7. Programme und der Editor
8. Ein Programm ausführen und fortsetzen
9. Analyse und Diagnose
10. Ein Programm im Zwilling ausführen
11. Prüfung, Berichte und Einrichteblätter
12. Eine echte Maschine verbinden
13. Der KI-Assistent
14. Sprache und Einheiten
15. Tastatur und Barrierefreiheit
16. Fehlerbehebung

Operations manual

1. Systemvoraussetzungen
2. Installieren und Entfernen
3. Lizenzen, Arbeitsplätze und Aktivierung
4. Ihre Daten und Backups
5. Studio aktualisieren
6. Datenschutz und was Studio sendet
7. Labore, Unterrichtsräume und gemeinsam genutzte Computer

Technical manual

1. Die Modelle des Zwillings und ihre Grenzen
2. Twinwrights eigener Kernel
3. G-Code und M-Code
4. CAM-Strategien und -Parameter
5. Passungen, Rundlauffehler und Prüfmethoden
6. Maschinenprofile
7. MC-1-Hardware

Reference

1. MCP tool reference
2. Einstellungsreferenz
3. Alarmcodes
4. Fehlercodes
5. Command line
6. Machine profile schema
7. Protokoll
8. Dateiformate
9. Sender-Software mit Twinwrights eigenem Kernel

User manual

Erste Schritte

Twinwright Studio betreibt einen digitalen Zwilling Ihrer CNC-Maschine. Sie spannen das Teil auf, erstellen oder öffnen ein Programm und führen es im Zwilling aus, bevor die reale Maschine zerspant. Dieses Kapitel führt Sie in etwa zehn Minuten von der Installation bis zu einem ersten Teil.

Installieren

Studio braucht 64-Bit-Windows 10 oder 11 (siehe Systemvoraussetzungen).

1. Starten Sie `TwinwrightStudio-<version>-setup.exe`. Sie brauchen keine Administratorrechte: Studio wird für Sie installiert.
2. Lesen Sie die Lizenzvereinbarung und klicken Sie auf **I Agree**.
3. Behalten Sie den Ordner bei oder wählen Sie einen anderen.
4. Lassen Sie auf der letzten Seite **Run Twinwright Studio** angehakt und klicken Sie auf **Finish**.

Im Startmenü steht Studio als **Twinwright Studio**. Die Installation auf vielen Computern und was wohin kommt: Installieren und Entfernen.

Der erste Start

Studio öffnet sich im Arbeitsbereich **Ausführen** mit eingeschalteter Maschine **MC-1 · Meine Werkbank**. Es werden keine Fragen gestellt.

Die Maschinenleiste zeigt **ALARM:11** und **Nächster Schritt: Referenzfahrt**. Das ist normal: Wie eine reale Maschine weiß der Zwilling erst nach der Referenzfahrt, wo seine Achsen stehen.

Was das Fenster zeigt

Eine neue Installation zeigt alles, was Ingenieurinnen und Ingenieure nutzen: alle sechs Arbeitsbereiche, die Kurven und die Einfärbungen. Wenn Sie nur Programme ausführen oder unterrichten, wählen Sie eine Stufe, die weniger zeigt: **Extras > Bedienstufe** (siehe Bedienstufen).

Sprache und Einheiten

Studio startet in der Sprache Ihres Systems, sofern Studio sie anbietet (Englisch, Chinesisch, Französisch, Deutsch, Japanisch oder Spanisch), und sonst auf Englisch. Die Einheiten und die Darstellung der Zahlen richten sich nach Ihrer Region.

Um sie zu ändern: **Extras > Language / 语言** und **Extras > Einheiten**. Siehe Sprache und Einheiten.

Editionen und die Testphase

Ihre Edition steht in der oberen Leiste, zum Beispiel **Learning**. Um sie anzuzeigen, klicken Sie darauf oder wählen Sie **Hilfe > Lizenz...**.

Edition	Fügt hinzu
Learning	Taschen, Konturen, Bohren, Gravieren und 3D-Parallelschlichten; den ganzen Zwilling, den Editor und die KI
Maker	Adaptives Ausräumen, alle 3D-Schlichtstrategien, Restmaterialbearbeitung, zwei Seiten und mehrere Aufspannungen
Pro	Verschachtelung, den Programmoptimierer, Berichte (den Prüfbericht, das Einrichteblatt) und das Prüfen mit dem Messtaster im Abgleich mit der Konstruktion

Jede Edition enthält alles aus den Editionen, die in der Tabelle über ihr stehen. Eine Funktion einer höheren Edition erscheint mit dem Namen ihrer Edition. Klicken Sie darauf, um zu lesen, was sie tut, und um eine Testphase zu starten oder die Lizenz zu öffnen.

Wenn eine Testphase oder eine Lizenz endet, ist Studio wieder die Edition Learning. Ihre Projekte, Programme und der G-Code, den Sie erzeugt haben, funktionieren weiter.

Eine Testphase starten

Sie können die Edition Maker und die Edition Pro je 30 Tage lang testen, einmal pro Computer.

1. Klicken Sie auf eine Funktion dieser Edition, zum Beispiel **Optimieren** im Arbeitsbereich **Programm**.
2. Klicken Sie auf **30 Tage testen...**
3. Klicken Sie auf **Testphase starten**.

Studio fragt beim Lizenzserver von Twinwright einmal die Testphase an und sendet dabei die Edition, einen Hash des Fingerabdrucks dieses Computers und die Studio-Version. Wenn die Testphase endet, steht das in einem Hinweis im Reiter **Meldungen**.

Eine Lizenz aktivieren

Öffnen Sie **Hilfe > Lizenz...**.

So aktivieren Sie online:

1. Geben Sie den Aktivierungscode ein (`TW-XXXXX-XXXXX-XXXXX-XXXXX`). Groß- und Kleinschreibung, Leerzeichen und Bindestriche spielen keine Rolle.
2. Klicken Sie auf **Aktivieren**. Ergebnis: **Aktiviert.**

So aktivieren Sie einen Computer ohne Internet:

1. Kopieren Sie unter **Offline aktivieren…** den Anforderungscode dieses Computers.
2. Öffnen Sie auf einem Computer mit Internet die Lizenzseite, fügen Sie den Anforderungscode ein und laden Sie die Lizenzdatei (`.twlic`) herunter.
3. Bringen Sie die Datei zurück und klicken Sie auf **Lizenzdatei importieren…**

Vor dem Ablauf verlängern verlängert die Lizenz online, bevor sie abläuft. Dabei werden nur die Lizenznummer, ein Hash des Fingerabdrucks dieses Computers und die Studio-Version gesendet.

Arbeitsplätze, die Übertragung auf einen anderen Computer und Unterrichtsräume: Lizenzen.

Ein erstes Teil in zehn Minuten

Das erste Teil entsteht am schnellsten mit einem der Beispielteile: Studio richtet das Rohteil, die Werkzeuge und den Werkstücknullpunkt für Sie ein.

Die Maschine öffnen

Die Maschine ist bereits geöffnet: **MC-1 · Meine Werkbank**, im Arbeitsbereich **Ausführen**. Andere Maschinen finden Sie unter **Datei > Maschinenbibliothek…**.

Einschalten und referenzieren

Drücken Sie **Referenzfahrt** (F3) oder die Taste, die die Maschinenleiste vorschlägt.

Ergebnis: Die drei Achsen werden nacheinander referenziert, in etwa 20 s. Die Maschinenleiste zeigt **Kein Alarm**.

Zeigt die Maschinenleiste **Not-Halt**, entriegeln Sie ihn zuerst (F1) und drücken Sie dann **Einschalten** (F2) und **Reset** (F12).

Ein Beispielprogramm öffnen

1. Wählen Sie **Datei > Beispielteile…**
2. Wählen Sie **Motorplatte für NEMA 17** (e01_motor_mount): eine Passbohrung, Flachsenkungen und Langlöcher, an Haltestegen freigeschnitten. Ergebnis: Studio spannt das Rohteil auf, lädt die Werkzeuge und das Programm und setzt den Werkstücknullpunkt auf das Rohteil. Die Konsole meldet, wann das Beispiel bereit ist.

Im Zwilling ausführen

1. Drücken Sie **Zyklusstart** (F5).
2. Wenn Sie es schneller ablaufen sehen möchten, wählen Sie **Simulation > Geschwindigkeit > ×10 Echtzeit**.

Die 3D-Ansicht zeigt, wie das Werkzeug das Rohteil zerspant. **Vorschub halt** (F6) unterbricht das Programm, **Zyklusstart** setzt es fort.

Das Ergebnis ansehen

- Ziehen Sie in der 3D-Ansicht, um sie zu drehen; das Mausrad zoomt.
- **Ansicht > Einfärbung > Werkstück: Abweichung vom Bauteil** färbt das Restmaterial und jeden Überschnitt ein.
- **Analyse** (Alt+5) zeigt, welche Sätze die Zeit gebraucht haben, und die Schätzungen des Zwillings für Schnittkraft und Rattern.

Das Rohteil einrichten

Für Ihr eigenes Teil richten Sie die Maschine in **Vorbereiten** (Alt+1) ein, statt ein Beispiel zu laden:

1. Gehen Sie zum Schritt **Aufspannung**.
2. Wählen Sie unter **Werkstück und Aufspannung** die **Form**, den **Werkstoff** und das **Spannmittel** und geben Sie die Maße ein.
3. Klicken Sie auf **Aufspannen**.

Siehe Einen Auftrag einrichten.

Das Werkzeug vermessen

1. Gehen Sie zum Schritt **Werkzeug**, wählen Sie das Werkzeug und klicken Sie auf **Werkzeug wechseln**.
2. Klicken Sie auf **Referenzwerkzeug messen**.

Die Maschine muss referenziert sein und die Spindel stillstehen.

Den Werkstücknullpunkt setzen

1. Gehen Sie zum Schritt **Nullpunkt** und fahren Sie das Werkzeug im Tippbetrieb an den Punkt, der Ihr Nullpunkt werden soll.
2. Klicken Sie auf **Alle nullen** oder nullen Sie die Achsen einzeln.

Mit einem Messtaster ermittelt die Messtasterkarte den Nullpunkt für Sie (siehe Antasten).

Öffnen Sie dann Ihr Programm (**Datei > Programm öffnen…**) und führen Sie es wie oben aus.

Wie es weitergeht

- Das Fenster und die Arbeitsbereiche: was wo ist.
- Programme erstellen: CAM in Studio.
- Ein Programm im Zwilling ausführen: was der Zwilling prüft und schätzt.
- Ein Programm ausführen und fortsetzen: Läufe, Alarme und erneutes Starten.

Ein **?** an einem Panel öffnet dieses Handbuch beim Kapitel zu diesem Panel. **Hilfe > Dokumentation** öffnet es am Anfang.

Das Fenster und die Arbeitsbereiche

Das Fenster von Studio hat oben die Menüs und eine obere Leiste, in der Mitte sechs Arbeitsbereiche und unten die Maschinenleiste. Jeder Arbeitsbereich hat eine Hauptaufgabe. Die Maschinenleiste zeigt Ihnen überall den Zustand der Maschine und den nächsten Schritt.

Die Teile des Fensters

Von oben nach unten:

- die **Titelzeile**: links die Menüs, in der Mitte der Name des Fensters, rechts die Tasten des Fensters;
- die **obere Leiste**: die Maschine, die Reiter der Arbeitsbereiche, die Edition, der Zustand der KI, die Bedienstufe und **?**;
- der **Arbeitsbereich**: die 3D-Ansicht und die Panels des gewählten Arbeitsbereichs;
- das **Dock** unter der 3D-Ansicht, in den Arbeitsbereichen, die eines haben;
- die **Maschinenleiste**, in jedem Arbeitsbereich dieselbe.

Studio hat keine Statusleiste: Die Maschinenleiste übernimmt diese Aufgabe.

Die Titelzeile und die Menüs

Die Menüs sind **Datei**, **Bearbeiten**, **Ansicht**, **Maschine**, **Programm**, **Simulation**, **KI**, **Extras** und **Hilfe**.

- Drücken Sie Alt und den Buchstaben eines Menüs, um es zu öffnen: Alt+D öffnet **Datei**.
- Drücken Sie Alt allein, um die Menütitel hervorzuheben, und bedienen Sie sie dann mit den Pfeiltasten, der Eingabetaste und Esc.

Solange ein Menü offen ist, erreicht keine Taste die Maschine. Ein Menü zeigt nur die Einträge Ihrer Bedienstufe (siehe Bedienstufen).

Die obere Leiste

Von links nach rechts:

- das Markenzeichen: Es öffnet **Datei > Maschinenbibliothek…**;
- die Maschine: ihr Modell, ihr Name und ihr Zustand. Klicken Sie darauf oder drücken Sie Strg+Umschalt+M, um die Maschine zu wechseln: Geben Sie etwas ein, um zu filtern, wählen Sie mit ↑ ↓ und bestätigen Sie mit der Eingabetaste;
- die Reiter der Arbeitsbereiche (siehe Die sechs Arbeitsbereiche);
- die Edition, zum Beispiel **Learning**: Ein Klick darauf öffnet **Hilfe > Lizenz…**;
- der Zustand der KI: **KI aus**, **KI ein**, **KI verbunden** oder **KI wartet auf Ihr OK**. Ein Klick darauf öffnet die Einstellungen der KI (siehe Der KI-Assistent);
- die Bedienstufe: Klicken Sie darauf, um eine andere Stufe zu wählen;
- **?**: die Liste der Tastenkombinationen.

Die Maschinenleiste

Die Maschinenleiste verläuft am unteren Rand des Fensters. Von links nach rechts:

1. **Der Zustand**: **Bereit**, **Läuft**, **Tippbetrieb**, **Referenzfahrt**, **Halt**, **Tür**, **Werkzeugwechsel**, **Prüfmodus**, **Ruhezustand**, **Alarm**, **Not-Halt** oder **Aus**.
2. **Das Programm**: sein Name, der Satz, an dem es steht, der Abschnitt, in dem es steht, ein Fortschrittsbalken sowie die gelaufene Zeit und die Restzeit. Ohne Programm zeigt dieser Teil **Kein Programm geladen**, ob die Maschine referenziert ist, das Werkzeug und die Nullpunktverschiebung.
3. **Der nächste Schritt**: **Kein Alarm** in Grün oder **Nächster Schritt:** und die Taste, die ihn ausführt. Bei einem Alarm zeigt er den Alarm, den nächsten Schritt und **Alle Schritte**; das listet die Schritte zurück zur Arbeit auf (siehe Alarme und was zu tun ist).
4. **Die Uhr**: die Zeit der Simulation, ihre Geschwindigkeit und die tatsächlich erreichte Geschwindigkeit. Die Sanduhrtaste hält die Simulation an.
5. **Die Tasten**: **Einzelsatz**, **Zyklusstart** (F5), **Vorschub halt** (F6) und **Stopp** (F7). Die Taste, die Studio vorschlägt, ist gefüllt; eine Taste, die Sie jetzt nicht benutzen können, ist abgeblendet.
6. **Not-Halt** (F1), die runde rote Taste am rechten Ende.

In einem schmalen Fenster lässt die Leiste zuerst die Zeit der Uhr weg, dann den Abschnitt und die Zeiten des Programms (sie wandern in den Tooltip), dann die Geschwindigkeiten der Uhr und zuletzt den Text des nächsten Schritts.

Die 3D-Ansicht

Die 3D-Ansicht zeigt die Maschine, das Werkstück und das Werkzeug, wie sie jetzt sind.

- Ziehen Sie mit der linken Maustaste, um die Ansicht zu drehen, und mit der rechten oder mittleren Maustaste, um sie zu verschieben. Das Mausrad zoomt zum Zeiger hin.
- Die Tasten oben links wählen eine Ansicht: isometrisch, vorn, rechts oder oben. **Werkzeug folgen** hält das Werkzeug in der Mitte. **Anzeige** schaltet die Spur der Werkzeugspitze und den Programmweg ein und aus.
- **Werkzeug-Nahansicht** zeigt das Werkzeug am Schnitt in einem kleinen Bild oben rechts. Sie erscheint im Arbeitsbereich **Ausführen**, solange ein Programm läuft.
- Die Chips am unteren Rand nennen das Werkstück und das Werkzeug.
- Klicken Sie auf das Werkstück oder die Spur: Die Leiste unter der Ansicht sagt Ihnen, welcher Programmsatz diesen Punkt erzeugt hat, und die Programmliste springt zu diesem Satz.

Dieselben Auswahlmöglichkeiten finden Sie unter **Ansicht > 3D-Ansicht**. **Ansicht > Einfärbung** färbt die Spur und das Werkstück nach Vorschub, Schnittkraft, Spanungsdicke, Ratterkennwert oder Programmsatz (ab Stufe Ingenieur).

Das Dock

Das Dock sitzt unter der 3D-Ansicht. Seine Reiter hängen vom Arbeitsbereich ab:

Arbeitsbereich	Reiter
Ausführen	**Programm**, **Kurven**, **Konsole**, **Meldungen**, **KI-Aktivität**
Vorbereiten	**Messergebnisse** (sobald ein Teil gemessen wurde), **Werkzeugtabelle**
CAM	**Prüfergebnisse**, **KI-Aktivität**, **Versionen**, **Programmvorschau**
Programm	**Konsole**

Analyse, **Diagnose** und der Werkstattmodus haben kein Dock.

- **Programm**: die Programmliste. **Aktuellem Satz folgen** hält den Satz, der gerade läuft, im Blick; die Zeitbalken zeigen die Zeit jedes Satzes in diesem oder im letzten Lauf.
- **Kurven**: die Messwerte des Laufs über die Zeit.
- **Konsole**: was Studio und die Steuerung gesagt haben. Geben Sie in der Zeile am unteren Rand eine Zeile G-Code oder einen `$`-Befehl ein und drücken Sie die Eingabetaste, um die Eingabe zu senden; ↑ und ↓ holen die vorherigen Zeilen zurück.
- **Meldungen**: Warnungen und Befunde, nach ihrer Herkunft gruppiert. Eine rote Zahl auf dem Reiter zählt die Warnungen.
- **Werkzeugtabelle**: die Werkzeuge der Maschine.
- **Messergebnisse**: das zuletzt mit dem Messtaster gemessene Teil, Prüfung für Prüfung – Soll, Toleranz, Messwert, Abweichung und die Schätzung des Zwillings (siehe Prüfung).
- **KI-Aktivität**: was KI-Clients auf den Maschinen dieses Computers getan haben.

Drücken Sie Strg+J oder klicken Sie auf den Pfeil am rechten Ende der Reiter, um das Dock auf seine Reiterzeile einzuklappen.

Die sechs Arbeitsbereiche

Wechseln Sie die Arbeitsbereiche mit den Reitern in der oberen Leiste, mit Alt+1 bis Alt+6, mit Strg+Tab und Strg+Umschalt+Tab oder über das Menü **Ansicht**. Wenn Sie in **CAM** oder **Programm Zyklusstart** drücken, gelangen Sie zu **Ausführen**. Solange ein Programm läuft, hat der Reiter **Ausführen** einen grünen Punkt.

Vorbereiten

Vorbereiten (Alt+1) macht die Maschine und das Teil Schritt für Schritt startklar: **Einschalten**, **Referenzfahrt**, **Aufspannung**, **Werkzeug**, **Nullpunkt**, **Programm** und **Fräsen**. Das Panel rechts folgt dem nächsten Schritt, bis Sie einen Schritt wählen. Siehe Einen Auftrag einrichten; zu den Maschinen, zum Einschalten und zur Referenzfahrt: Maschinen und Einschalten.

CAM

CAM (Alt+2) erstellt Programme: links die Aufspannung, die Formen und die Bearbeitungen des Auftrags, rechts die Felder des gewählten Eintrags, unten **Programm generieren**. Siehe Programme erstellen. Ab Stufe Ingenieur.

Programm

Programm (Alt+3) enthält die Programme und die Beispielteile, den Editor und eine Vorschau mit den Prüfungen des Programms. Siehe Programme und der Editor.

Ausführen

Ausführen (Alt+4) führt das geladene Programm aus. Die rechte Spalte zeigt die Position, die Spindel und den Vorschub, die Overrides, den Schnitt, die Tasten des Bedienfelds, den Handbetrieb sowie die Werkzeuge und den Nullpunkt. Siehe Ein Programm ausführen und fortsetzen.

Analyse

Analyse (Alt+5) blickt auf Läufe zurück: ihre Kurven, die Zeit und die Kräfte jedes Satzes und die Stabilitätskarte. Ab Stufe Ingenieur. Siehe Analyse und Diagnose.

Diagnose

Diagnose (Alt+6) zeigt die Versorgungskette der Maschine, die Eingänge der Steuerung und die Antriebe und lässt Sie Fehler in den Zwilling einspeisen. Ab Stufe Techniker. Siehe Analyse und Diagnose.

Meldungen und was als Nächstes zu tun ist

Studio sagt Ihnen an diesen Stellen, was passiert ist und was zu tun ist:

- die Maschinenleiste: der nächste Schritt oder der Alarm und die Schritte zurück zur Arbeit;
- der Reiter **Meldungen** im Dock: Warnungen und Befunde;
- die **Konsole**: jede Zeile der Steuerung und von Studio;
- **Extras > Prüfprotokoll…** (Strg+L): alles, mit Angabe, wer es wann getan hat.

Maschine > Nächsten Schritt ausführen führt den Schritt aus, den die Maschinenleiste zeigt, und nur diesen. Hat ein Alarm ein Programm gestoppt, enden die Schritte mit dem Satz, ab dem neu gestartet wird (siehe Ab einem Satz neu starten).

Bedienstufen

Eine Bedienstufe zeigt, was ein Mensch in dieser Phase braucht, und blendet den Rest aus. Es gibt fünf:

Stufe	Kommt hinzu
Demo	Der Arbeitsbereich **Ausführen** mit den Beispielteilen; die Reiter **Programm** und **Meldungen**
Bediener	**Vorbereiten** und **Programm**; die Overrides, das Bedienfeld, der Handbetrieb, die Werkzeuge und der Nullpunkt; die Untermenüs von **Maschine**; die Konsole, die Werkzeugtabelle, **KI-Aktivität**; die Einstellungen, das Prüfprotokoll, das Menü **KI**; das Wechseln der Maschine
Techniker	**Diagnose**; das Anordnen des Werkstücks; **Werkseinstellungen…**
Ingenieur	**CAM** und **Analyse**; die Schnittwerte, die Kurven, **Einfärbung**, der Export von Kurven, das Bauteil zum Vergleich
Forscher	Die Karte **Modell und Daten** in **Diagnose**

Jede Stufe hat alles, was die Stufen über ihr in der Tabelle haben. Eine neue Installation beginnt auf der Stufe **Ingenieur**.

Um die Stufe zu ändern, verwenden Sie **Extras > Bedienstufe**, die Stufe in der oberen Leiste oder **Extras > Einstellungen… > Allgemein**.

> **Hinweis**
>
> Eine Stufe richtet das Fenster zum Lehren und Lernen ein. Sie sperrt nichts: Die Tasten der Maschine funktionieren auf jeder Stufe.

Hell, Dunkel und Werkstattmodus

Ansicht > Darstellung wählt **Wie im System**, **Hell** oder **Dunkel** und schaltet **Werkstattmodus (groß)** ein oder aus. Die beiden Auswahlen sind voneinander unabhängig: Der Werkstattmodus funktioniert in jeder Darstellung.

Die eigenen Panels von Studio ändern sich sofort. Listen und Textfelder, die das System zeichnet, übernehmen das neue Aussehen beim nächsten Start.

Der Werkstattmodus ist für die Arbeit an der Maschine auf Armlänge gedacht:

- die Maschinenleiste, ihre Tasten und der Not-Halt sind größer, ebenso der Text und die Programmliste;
- **Ausführen** hat ein eigenes Layout: links die Position, das Programm und eine große Programmliste, unter der 3D-Ansicht die Anzeigen, rechts die Overrides, die Tasten des Bedienfelds und die Meldungen, kein Dock;
- die 3D-Ansicht hat zwei Ansichtstasten und keine Werkzeug-Nahansicht.

Panels und ihre Größen

Ziehen Sie die Kante zwischen zwei Panels, um deren Größen zu ändern; ein Doppelklick auf die Kante stellt die Standardgröße wieder her. Die 3D-Ansicht behält bei 100 % Anzeigeskalierung mindestens 360 × 240 Pixel. Die Kanten der Panels lassen sich nur mit der Maus verschieben. **Ansicht > Layout zurücksetzen** setzt jedes Panel aller Arbeitsbereiche auf einmal auf seine Entwurfsgröße zurück.

Studio merkt sich für jede Maschine die Größen der Panels, den Reiter des Docks und ob es eingeklappt ist sowie den Arbeitsbereich, in dem Sie waren. Die Position und die Größe des Fensters merkt es sich für den nächsten Start.

Maschinen und Einschalten

Die Maschinen in Studio, wie Sie zwischen ihnen wechseln und wie eine Maschine eingeschaltet und referenziert wird, bevor sie arbeitet.

Maschinen

Jede Maschine behält ihre eigene Welt: ihre Position, ihr Werkstück, ihre Werkzeuge und Nullpunktverschiebungen, ihre Temperaturen und ihre Betriebsstunden. Die Maschine in Ihrem Fenster ist die, die läuft; die anderen warten so, wie sie gespeichert wurden. In 1.0 läuft immer nur eine Maschine.

Maschinen wechseln

1. Klicken Sie in der oberen Leiste auf die Maschine oder drücken Sie Strg+Umschalt+M (**Datei > Maschine wechseln…**).
2. Geben Sie etwas ein, um eine Maschine zu finden, wählen Sie sie mit ↑ ↓ und drücken Sie die Eingabetaste. Ergebnis: Studio speichert die Maschine, die Sie verlassen, und startet mit der anderen neu. Diese Maschine startet kalt: Schalten Sie sie ein und referenzieren Sie sie.

Während ein Programm läuft, können Sie nicht wechseln. Die Liste zeigt auch die **Übungsmaschinen**, die KI-Clients im Hintergrund verwenden; **Ansehen (schreibgeschützt)** zeigt sie in **KI-Aktivität**.

Die Maschinenbibliothek

Datei > Maschinenbibliothek… oder ein Klick auf das Markenzeichen zeigt Ihre Maschinen mit Zustand, Werkstück und Werkzeug sowie die Maschinenmodelle. Ein weiterer Klick auf das Markenzeichen führt zurück.

Für die ausgewählte Maschine:

- **Öffnen**: zu ihr wechseln.
- **Umbenennen…**: ihr einen eigenen Namen geben.
- **Als neue Maschine kopieren…**: eine neue Maschine mit der Welt dieser Maschine, so wie sie zuletzt gespeichert wurde.
- **Ordner anzeigen**: ihre Welt auf der Festplatte.
- **Auf Werkszustand zurücksetzen… (zwei Klicks innerhalb von 5 s)**: die Maschine, wie sie geliefert wurde (siehe Eine Maschine neu aufsetzen).

Neue Maschine… fügt eine Maschine des Modells im Werkszustand hinzu. Neue Maschinen, Kopien und das Zurücksetzen auf den Werkszustand brauchen die Stufe Techniker. Die Zeile **Reale MC-1** ist ein Platzhalter für später: In 1.0 wird eine reale Maschine über die Kommandozeile verbunden (siehe Eine reale Maschine).

Einschalten und Referenzfahrt

Einschalten

Der Zwilling hat die Versorgungskette der MC-1: Der Hauptschalter QF1 versorgt die Steuerung mit 24 V, und **Einschalten** schließt das Schütz KM1, das die Antriebe mit 48 V versorgt und den Spindelantrieb mit Strom speist. Der Not-Halt öffnet KM1.

1. Gehen Sie zum Schritt **Einschalten**.
2. Ist der Hauptschalter aus, klicken Sie auf **Hauptschalter ein**.
3. Zeigt die Taste **Gedrückt**, ist der Not-Halt gedrückt: Klicken Sie darauf oder drücken Sie F1, um ihn zu entriegeln.
4. Klicken Sie auf **Einschalten** (F2). Ergebnis: Das Panel zeigt 48 V eingeschaltet und jeden Antrieb **Bereit**. Der nächste Schritt ist **Referenzfahrt**.

Referenzfahrt

Klicken Sie auf **Referenzfahrt** (F3). Zuerst fährt Z nach oben, dann X und Y zusammen, jede Achse bis zum Schalter am Ende ihres Verfahrwegs; in Echtzeit dauert das etwa 20 s.

Ergebnis: **Referenziert: X Y Z**, und die Software-Endschalter sind aktiv. Bis die Maschine nach dem Einschalten referenziert ist, steht die Steuerung in ALARM:11, und der Werkzeuglängensensor, der Messtaster und **Nullpunkt anfahren** warten.

Software-Endschalter und der Verfahrweg

Die MC-1 verfährt 200 mm in X, 150 mm in Y und 80 mm in Z. Nach der Referenzfahrt ist die Maschinenkoordinate jeder Achse an ihrem Referenzende 0: X reicht von −200 bis 0, Y von −150 bis 0 und Z von −80 bis 0, mit 0 am oberen Ende von Z.

- Eine Tippbewegung, die über eine Grenze hinausginge, wird nicht ausgeführt (error:15). Eine gehaltene Tippbewegung stoppt kurz davor.
- Ein Programmsatz, der über eine Grenze hinausginge, stoppt das Programm mit ALARM:2. Die Position bleibt erhalten: Klicken Sie auf **Reset** und **Entsperren** und prüfen Sie dann das Programm und den Werkstücknullpunkt.
- Eine Achse, die einen Endschalter erreicht, löst ALARM:1 aus: Klicken Sie auf **Reset** und **Entsperren**, fahren Sie im Tippbetrieb vom Schalter weg und klicken Sie dann auf **Referenzfahrt**.

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Einen Auftrag einrichten

Der Arbeitsbereich **Vorbereiten** macht die Maschine und das Teil Schritt für Schritt startklar: das Einschalten, die Referenzfahrt, das Rohteil und wie es gespannt wird, die Werkzeuge und der Werkstücknullpunkt. Jeder Schritt erscheint zuerst im Zwilling, sodass Sie die Aufspannung sehen, bevor sich etwas bewegt. Die Maschinen, das Einschalten und die Referenzfahrt haben ein eigenes Kapitel: Maschinen und Einschalten.

Der Arbeitsbereich Vorbereiten

Vorbereiten (Alt+1) hat links die Schritte, in der Mitte die 3D-Ansicht und rechts das Panel des gewählten Schritts. Das Dock unter der Ansicht enthält die **Werkzeugtabelle**.

Die Schritte

Schritt	Sein Panel	Erledigt, wenn
Einschalten	Der Hauptschalter, **Einschalten**, **Reset**, **Entsperren**	Die Steuerung läuft und die Antriebe haben Spannung
Referenzfahrt	**Referenzfahrt**, **Entsperren**, **Reset**	Alle drei Achsen sind referenziert
Aufspannung	Das Rohteil, wie es gespannt wird und wo	Ein Rohteil ist aufgespannt oder der Tisch bleibt leer
Werkzeug	Das Werkzeug in der Spindel, der Werkzeuglängensensor	Das Referenzwerkzeug ist vermessen, eine Werkzeugtabelle ist geladen oder der Werkstück-Z-Nullpunkt ist gesetzt
Nullpunkt	Tippbetrieb, der Werkstücknullpunkt, die Messtasterkarte	Eine Nullpunktverschiebung ist gesetzt
Programm	**Programm öffnen…**, **Beispielteile…**, **Programm prüfen**	Ein Programm ist geladen
Fräsen	**Zyklusstart**, **Einzelsatz**	Das Programm ist bis zum Ende gelaufen

Ein erledigter Schritt hat einen grünen Haken; der nächste Schritt hat an seinem Ende einen kleinen Pfeil. Das Panel zeigt den nächsten Schritt, bis Sie einen anderen anklicken; klicken Sie auf den nächsten Schritt, damit das Panel wieder folgt. Die Zeile unter den Schritten sagt, was jetzt zu tun ist. Das **?** auf einem Panel öffnet seinen Teil dieses Handbuchs.

Die 3D-Ansicht beim Vorbereiten

Solange Sie die Felder des Rohteils ändern, wird das Rohteil, das Sie einrichten, durchsichtig auf dem Tisch gezeichnet: grün, wenn es so aufgespannt werden kann, rot, wenn nicht. **Aufspannen** setzt es dorthin; **Zurücksetzen** oder Esc verwirft die Änderung.

Das Rohteil

Form, Werkstoff und Maße

Im Schritt **Aufspannung**, auf der Karte **Werkstück und Aufspannung**:

1. Wählen Sie die **Form**: **Quader**, **Rundstab (stehend)**, **Rohr (stehend)**, **Sechskantstab (stehend)**, **Rundstab (liegend)** oder **STL-Datei (Form importieren)…**.
2. Wählen Sie den **Werkstoff** oder geben Sie seinen Namen aus einer anderen Norm ein (C45, EN AW-6061, SUS304): Die Liste zeigt die Werkstoffsorte, deren Schnittdaten Studio übernimmt, und die Aufspannung behält den Namen, den Sie eingegeben haben.
3. Geben Sie die Maße ein, die die Form braucht: **Länge L**, **Breite W**, **Höhe H**, **Durchmesser D** und so weiter.
4. Wählen Sie, wie es gespannt wird und wo (unten), und klicken Sie dann auf **Aufspannen**. Ergebnis: Das Panel nennt, was auf dem Tisch liegt, und seine Masse.

Die Maße stehen in der Anzeigeeinheit; in Zoll nehmen die Felder auch Brüche an (siehe Zollbrüche).

> **Hinweis**
>
> Ein Name aus einer anderen Norm ist eine ähnliche Sorte: Die Schnittdaten stammen von der nächstliegenden Sorte der Liste. Welcher Werkstoff tatsächlich vorliegt, steht im Zeugnis des Lieferanten.

Ein Rohteil aus einer STL-Datei

Wählen Sie in der Formliste **STL-Datei (Form importieren)…** und dann die Datei aus (in mm). Studio setzt die Form auf ihrem tiefsten Punkt ab und zentriert sie; der umschließende Quader ergibt die Maße. Material unter einem Überhang gilt als massiv. Lässt sich die Datei nicht lesen, sagt die Konsole, warum.

Das Rohteil spannen

Wählen Sie das **Spannmittel**: **Spannpratzen · beide Enden**, **Spannpratzen · beide Seiten**, **Spannpratzen · vier Seiten**, **Schraubstock**, **Prisma (liegende Stange)**, **Doppelseitiges Klebeband**, **Nicht gespannt (nur aufgelegt)** oder **Ideal gespannt (für Skripte)**.

Das Rohteil und sein Spannmittel müssen auf dem Tisch (300 × 220 mm) bleiben und vom Werkzeuglängensensor hinten links frei bleiben. Während ein Programm läuft, vergleicht der Zwilling die Schnittkraft mit dem, was das Spannmittel hält: Er warnt, wenn die Reserve klein ist, und ein Rohteil, das rutscht, bewegt sich im Zwilling, sodass das danach Gefräste nicht mehr stimmt.

Schraubstock und Parallelunterlagen

Überstand über Backen ist, wie weit die Oberseite des Rohteils über die Backen hinausragt: 5 mm, solange Sie es nicht ändern. Studio wählt die Parallelunterlagen, die mindestens so viel ergeben. Der Schraubstock öffnet sich von 3,25 bis 77 mm, und die Backen müssen mindestens 2 mm des Rohteils fassen.

Spannpratzen

Spannpratzen auf Stiftschrauben in den Gewindebohrungen des Tisches (M6, im 25-mm-Raster): **beide Enden** entlang der Länge des Rohteils, **beide Seiten** quer dazu, **vier Seiten** beides. Jede Spannpratze braucht eine Bohrung 7 bis 35 mm außerhalb der Kante des Rohteils; gibt es keine, nennt das Panel die Spannpratze, die keine Bohrung findet. Verwenden Sie für eine liegende Stange das Prisma oder den Schraubstock.

Doppelseitiges Klebeband

Das Klebeband hält das Rohteil über seine Fläche auf dem Tisch. Verwenden Sie es für leichte Schnitte und flache Teile, nicht für eine liegende Stange. Sobald es loslässt, hält nur noch die Reibung das Rohteil. **Nicht gespannt (nur aufgelegt)** hält allein durch Gewicht und Reibung; **Ideal gespannt (für Skripte)** rutscht nie, für Skripte und Tests.

Platz über dem Rohteil

Die Programme, die das CAM schreibt, nutzen den Platz über dem Rohteil so:

- Ein Programm beginnt, endet und wechselt die Werkzeuge am oberen Ende des Z-Verfahrwegs.
- Zwischen den Bearbeitungen fährt das Werkzeug in der sicheren Höhe hinüber: 5 mm über der Oberseite des Rohteils oder 2 mm über einer Spannpratze, die höher steht als das Rohteil.
- Lässt die Aufspannung weniger Platz als das (ein langes Werkzeug, ein hoher Schraubstock, ein hohes Rohteil), senkt Studio die sichere Höhe auf das, was da ist, mindestens 0,3 mm unter dem oberen Ende des Verfahrwegs, und teilt das beim Programm mit. Wäre das weniger als 0,5 mm über dem Rohteil, schreibt es das Programm nicht und sagt, warum.

Die Tabelle **Z-Freiraum** des Einrichteblatts prüft den Messtaster und jedes Werkzeug für jede Aufspannung. Siehe Z-Freiraum.

Das Rohteil platzieren

Position X und **Y** legen die Mitte des Rohteils in den Tischkoordinaten des Zwillings fest (die Tischmitte liegt bei X 100, Y 75). **Winkel °** dreht es gegen den Uhrzeigersinn, von oben gesehen.

- **Aufspannen**: ein neues Rohteil dort. Das bisherige Rohteil und das, was darin gefräst wurde, sind weg.
- **Hierher versetzen**: dasselbe Rohteil, versetzt, gedreht oder anders gespannt. Das Gefräste bleibt erhalten.
- **Entnehmen**: das Rohteil vom Tisch nehmen.
- **Zurücksetzen**: die Felder zurück auf das, was auf dem Tisch liegt.

Das Aufspannen wartet, solange sich die Spindel dreht, die Maschine fährt oder ein Programm läuft.

In der 3D-Ansicht anordnen

Klicken Sie auf **Werkstück anordnen** (ab Stufe Techniker) oder verwenden Sie **Ansicht > 3D-Ansicht > Werkstück anordnen**. Danach:

- ziehen Sie das Rohteil, um es zu verschieben, den orangefarbenen Griff, um es zu drehen, und die blauen Griffe, um seine Maße zu ändern (mit der Umschalttaste in feinen Schritten, mit Strg in groben);
- oder verwenden Sie die Tasten: Die Pfeiltasten verschieben es um 1 mm (Umschalttaste 0,1 mm, Strg 10 mm), Q und E drehen es um 15° (Umschalttaste 1°), die Eingabetaste spannt es auf, Esc stellt es zurück.

Wenn Sie den Arbeitsbereich **Vorbereiten** verlassen, endet das Anordnen.

Das Rohteil wenden

Wenden dreht das Rohteil in seinem Spannmittel um 180° um seine Länge: Oben und unten tauschen die Plätze, ebenso vorn und hinten. Das Gefräste dreht sich mit. Die Nullpunktverschiebungen bleiben, wie sie waren; setzen Sie also den Werkstücknullpunkt neu (siehe Antasten).

Bei einem Teil, das von zwei Seiten bearbeitet wird, plant CAM das Wenden zwischen seinen Aufspannungen (siehe Aufträge mit mehreren Aufspannungen).

Werkzeuge und die Werkzeugtabelle

Die **Werkzeugtabelle** im Dock listet die Werkzeuge der Maschine: die T-Nummer, das Werkzeug und die Spezifikation, die das Programm liest. Das Werkzeug in der Spindel ist mit **In der Spindel** markiert. Die Tabelle kommt mit einem Beispielteil, einem CAM-Auftrag oder einem KI-Client; im Fenster können Sie sie nicht ändern.

Mit einer Werkzeugtabelle wechselt das Programm die Werkzeuge selbst (`M6`): Das Werkzeug wird eingesetzt, die Maschine vermisst es am Werkzeuglängensensor und macht weiter. Ohne Werkzeugtabelle wartet ein `M6`, bis Sie das Werkzeug einsetzen (**Werkzeug wechseln** im Schritt **Werkzeug**) und dann **Zyklusstart** drücken.

Das Werkzeug in der Spindel wechseln

1. Gehen Sie zum Schritt **Werkzeug**.
2. Wählen Sie das Werkzeug in der Liste **Werkzeug**, zum Beispiel **Schaftfräser Ø6, 2 Schneiden (Auskraglänge 15.5)**.
3. Klicken Sie auf **Werkzeug wechseln**.

Die Spindel muss stehen und die Maschine bereit sein. Ist das neue Werkzeug länger oder kürzer, vermessen Sie es (**Werkzeugkorrektur setzen**) oder setzen Sie den Werkstück-Z-Nullpunkt neu. Die Liste enthält die Standardwerkzeuge, zuerst die metrischen (zuerst die Zoll-Werkzeuge, wenn Studio Zoll anzeigt). Wie ein Werkzeug beschrieben wird, zum Beispiel `flat 6 n2 L15.5`: Werkzeuge und die Werkzeugtabelle.

Werkzeuge mit dem Werkzeuglängensensor vermessen

Der Werkzeuglängensensor steht hinten links am Tisch. Die Maschine muss referenziert und bereit sein, und die Spindel muss stehen.

1. Klicken Sie auf **Referenzwerkzeug messen**, während das Werkzeug in der Spindel steckt, mit dem Sie den Werkstück-Z-Nullpunkt setzen.
2. Klicken Sie nach jedem Werkzeugwechsel auf **Werkzeugkorrektur setzen**. Ergebnis: Studio vermisst das neue Werkzeug und setzt seine Längenkorrektur (`G43.1`) aus der Differenz.

Dieselben Tasten finden Sie unter **Maschine > Werkzeuglängensensor und Messtaster**. Die Referenz bleibt bis zum Schließen von Studio erhalten: Messen Sie sie nach einem Neustart erneut.

Das Bauteil zum Vergleich

Mit der Konstruktion des Bauteils zeigt der Zwilling, wo sich das gefräste Bauteil davon unterscheidet (ab Stufe Ingenieur).

1. Setzen Sie zuerst den Werkstücknullpunkt: Das Bauteil wird beim Laden am Nullpunkt des Programms platziert.
2. Klicken Sie im Schritt **Aufspannung** unter **Bauteilvergleich** auf **Bauteil-STL (Vergleich)…** und wählen Sie die STL-Datei des Bauteils (in mm, ihr Ursprung am Nullpunkt des Programms). Ergebnis: Das Werkstück wird nach seiner Abweichung vom Bauteil eingefärbt (**Ansicht > Einfärbung > Werkstück: Abweichung vom Bauteil**): Innerhalb von ±20 µm gilt als gut, die anderen Farben zeigen Restmaterial und Überschnitte.

Vergleich beenden entfernt das Bauteil. Die Beispielteile laden ihr eigenes.

Aufspannungen speichern und öffnen

- **Aufspannung speichern…** schreibt die Aufspannung der Karte (die Form, die Maße, den Werkstoff, das Spannmittel und den Ort) in eine JSON-Datei.
- **Aufspannung öffnen…** liest eine Aufspannung zurück in die Karte; klicken Sie auf **Aufspannen**, um sie einzurichten. Es öffnet auch eine STL-Datei als Form des Rohteils.
- **Werkstück exportieren…** speichert das Werkstück, wie es gefräst ist, als STL (in Programm-, Rohteil- oder Maschinenkoordinaten) oder als 16-Bit-Höhenkarte (PNG). Dasselbe steht im Menü Datei: **Datei > Werkstück exportieren…**, in jedem Arbeitsbereich.

Jede Maschine behält ihre Aufspannung und ihr gefrästes Werkstück in ihrer Welt. Studio speichert sie jedes Mal, wenn die Maschine nach einer Änderung zwei Sekunden stillsteht, und beim Schließen des Fensters; nach einem Absturz oder Stromausfall kommt die Maschine so zurück, wie sie zuletzt stillstand.

Aufträge mit mehreren Aufspannungen

Ein Teil, das von zwei Seiten bearbeitet oder auf zwei Arten gespannt wird, ist ein Auftrag mit mehreren Aufspannungen. Im Arbeitsbereich **CAM** fügt **Aufspannung…** dem Auftrag eine Aufspannung hinzu. Der Auftragsbaum hat dann für jede Aufspannung eine Ebene, mit ihren Bearbeitungen und ihrem Programm darunter, und zwischen zwei Aufspannungen einen Schritt **Wenden**. Ein Auftrag mit einer Aufspannung hat keine solche Ebene.

- Jede Aufspannung hat ihre eigenen Bearbeitungen und ihr eigenes Programm. Die Programme sind nach dem Auftrag und der Nummer der Aufspannung benannt, zum Beispiel `plate-1.nc` und `plate-2.nc`.
- Eine Aufspannung beginnt mit dem Teil, wie es die vorherigen Aufspannungen hinterlassen haben. Im Zwilling führt Studio die Aufspannung 1 aus, wendet das Teil, wie es der Schritt **Wenden** sagt, und schreibt das Programm der nächsten Aufspannung für dieses Teil. In der 3D-Ansicht wird das, was vorher gefräst wurde und jetzt unten liegt, gestrichelt gezeichnet.
- Der Baum nennt eine Aufspannung nach der Anzahl der Wendungen des Teils Vorderseite oder Rückseite.
- Unter dem Panel einer Bearbeitung zeigt **Diese Aufspannung** das Spannmittel der Aufspannung, ihren Nullpunkt, wie das Teil gewendet wurde und das Programm.
- **Prüfergebnisse** hat eine Zeile für jede Aufspannung und eine für das Wenden. **Gesamt** addiert die Zeit an der Maschine und nennt den schlechtesten Ratterkennwert und den größten Überschnitt. Es vergleicht auch das fertige Teil nach der letzten Aufspannung mit dem Bauteil laut Konstruktion.

Ein Auftrag hat höchstens 8 Aufspannungen. Ein verschachtelter Auftrag kann noch keine Aufspannungen haben. Mehrere Aufspannungen gibt es ab der Edition Maker (siehe Editionen).

Das Einrichteblatt hat für jede Aufspannung einen Abschnitt, mit dem, was dazwischen zu tun ist. Siehe Einrichteblätter.

Das Teil zwischen den Aufspannungen wenden

Wählen Sie im Auftragsbaum **Wenden**. Das Panel dazu sagt, wie das Teil zur nächsten Aufspannung kommt. Die Einstellungen gehören zur Aufspannung nach dem Schritt.

Einstellung	Auswahl
Wendeart	**Um U, 180°**: Das linke und das rechte Ende bleiben, wo sie sind; vorn und hinten tauschen die Plätze, ebenso oben und unten. **Um V, 180°**: Die vordere und die hintere Kante bleiben; links und rechts tauschen die Plätze, ebenso oben und unten. **Nicht gewendet (umgespannt)**: Das Teil bleibt, wie es liegt, und nur das Spannmittel ändert sich.
Nullpunkt	**Ecke vorn links oben** oder **Mitte der Oberseite**: wo der Werkstücknullpunkt wiedergefunden wird (siehe Den Werkstücknullpunkt übertragen)
Spannmittel	Wie das Teil ab dieser Aufspannung gehalten wird: dasselbe Spannmittel oder ein anderes

U verläuft längs der X-Achse der Maschine und V längs ihrer Y-Achse, durch das Teil, wie es in der vorherigen Aufspannung liegt.

Aufgaben des Bedieners listet die Schritte an der Maschine auf: das Programm der vorherigen Aufspannung zu Ende laufen lassen (Spindel gestoppt, Kühlung aus), das Teil ausspannen, wenden und wieder einspannen, den Werkstücknullpunkt neu setzen und das nächste Programm laden. **Im Zwilling** sagt, was der Zwilling mit dem Teil tut.

> **Achtung**
>
> Wenden Sie das reale Teil so, wie der Schritt es sagt. Das nächste Programm wird im Zwilling nur für das Teil geprüft, das auf diese Weise gewendet wurde. Ein Teil, das um die andere Achse oder gar nicht gewendet wird, wird dort gefräst, wo nichts geprüft wurde.

Den Werkstücknullpunkt übertragen

Wird das Teil gewendet oder in einem anderen Spannmittel gehalten, liegen seine Ecken nicht mehr dort, wo sie waren: Der Werkstücknullpunkt (G54) muss neu gefunden werden. Der Auftragsbaum sagt das neben der Aufspannung, und das Programm der Aufspannung beginnt an dem Nullpunkt, den Sie gewählt haben:

- **Ecke vorn links oben**: die vordere linke Ecke der Oberseite des Teils, wie das Teil jetzt liegt. Das ist der Nullpunkt nach einem Wenden oder einem Wechsel des Spannmittels, wenn Sie keinen anderen wählen.
- **Mitte der Oberseite**: die Mitte der Oberseite des Teils, zum Beispiel für ein rundes Teil.

Finden Sie den Nullpunkt mit dem Messtaster (siehe Antasten). Das Einrichteblatt nennt die Antastschritte für jede Aufspannung.

Wird das Teil weder gewendet noch anders gespannt, bleibt der Nullpunkt der vorherigen Aufspannung erhalten und wird nicht neu gefunden. So arbeitet eine zweite Aufspannung, die nur andere Werkzeuge verwendet; ihr Abschnitt im Einrichteblatt sagt, dass der Werkstücknullpunkt nicht erneut angetastet wird.

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Antasten und den Werkstücknullpunkt finden

Mit einem Messtaster in der Spindel findet Studio eine Kante, eine Ecke, die Mitte einer Bohrung oder eines Zapfens oder die obere Fläche und setzt daraus den Werkstücknullpunkt. Die Messtasterkarte prüft den Zyklus vor dem Start gegen die Aufspannung, und der Zwilling führt ihn so aus, wie es die Maschine täte.

Die Messtasterkarte befindet sich in **Vorbereiten**, Schritt **Nullpunkt**, unter den Tipptasten. In 1.0 arbeitet die Messtasterkarte am Zwilling. Tasten Sie an einer realen Maschine mit der eigenen Steuerung der Maschine an.

Vor dem Start

Sie brauchen:

- das Rohteil, aufgespannt auf dem Tisch (Schritt **Aufspannung**);
- den Messtaster in der Spindel (siehe Der Messtaster in der Werkzeugtabelle);
- die gestoppte Spindel.

Die Maschine muss nicht referenziert sein: Ein Zyklus, der die Maschine in einem Alarm oder nicht referenziert vorfindet, setzt sie zurück und referenziert sie zuerst, und die Karte sagt das.

> **Achtung**
>
> An der MC-1 lässt der Schraubstock mit dem 30-mm-Messtaster keinen Platz über dem Rohteil (siehe Platz über dem Rohteil). Fixieren Sie das Rohteil beim Antasten auf doppelseitigem Klebeband oder unter Spannpratzen.

Der Messtaster in der Werkzeugtabelle

1. Gehen Sie in **Vorbereiten** zum Schritt **Werkzeug**.
2. Wählen Sie in **Werkzeug** den **3D-Messtaster, Tastkugel Ø2 (Auskraglänge 30, zum Kantenantasten)**.
3. Klicken Sie auf **Werkzeug wechseln**.

Ergebnis: Die erste Zeile der Messtasterkarte, **In der Spindel**, nennt den Messtaster. Steckt etwas anderes in der Spindel, sagt die Karte, dass Sie zuerst den Messtaster einsetzen sollen.

In einer Werkzeugbeschreibung wird der Messtaster als `probe 2 L30` geschrieben: eine 2-mm-Kugel, die 30 mm aus dem Halter ragt.

Platz über dem Rohteil: das Z-Budget

Mit Z am oberen Ende des Verfahrwegs muss die Spitze des Messtasters über den höchsten Punkt der Aufspannung hinwegkommen: die Oberseite des Rohteils, die Backen des Schraubstocks oder die Spannpratzen. Studio berechnet das für das Werkzeug in der Spindel und für jedes Werkzeug der Werkzeugtabelle, sobald sich die Aufspannung oder die Werkzeuge ändern. Kommt ein Werkzeug nicht über die Aufspannung, sagt der Schritt **Aufspannung**, wie viel ihm fehlt, und schlägt ein kürzeres Werkzeug oder eine niedrigere Aufspannung vor (Spannpratzen statt Schraubstock).

Die Messtasterkarte prüft dasselbe für den Zyklus: Sie fährt den Messtaster in einer sicheren Höhe über dem Rohteil und dem Spannmittel hinüber und zeigt diese Höhe in den Prüfungen.

Die Messtasterkarte

Von oben:

- **In der Spindel**: der Messtaster und ob er kalibriert ist (**Kalibriert** mit der Uhrzeit oder **Nicht kalibriert**);
- **Messtaster-Zyklen**: die sechs Zyklen und die Felder des gewählten;
- die Prüfungen;
- **Weg in 3D zeigen** und **Antasten starten…**.

Nach einem Zyklus zeigt die Karte stattdessen ihr Ergebnis (siehe Den Werkstücknullpunkt aus dem Gefundenen setzen). Unter der Messtasterkarte befindet sich die Karte **Teil messen**, darunter die Karte **Berichte** (siehe Prüfung).

Zahlenfelder nehmen die Anzeigeeinheit an, in Zoll auch Brüche. Ein Feld, das sich nicht lesen lässt, wird rot und sagt, warum (siehe Zahlen eingeben).

Die Prüfungen

Die Karte prüft den Zyklus, während Sie ihn einrichten, bevor sich etwas bewegt. Jede Prüfung ist grün (✓), gelb (!) oder rot (✕). Jede rote Prüfung hält **Antasten starten…** grau.

Prüfung	Grün	Gelb oder rot
Der Messtaster	**Messtaster in der Spindel, nicht ausgelöst**	Kein Messtaster in der Spindel; der Eingang des Messtasters ist bereits ausgelöst
Die Maschine	**Spindel steht, Maschine referenziert und bereit**	Gelb: im Alarm oder nicht referenziert, der Zyklus referenziert zuerst. Rot: Die Spindel dreht sich, oder die Maschine ist beschäftigt
Die Software-Endschalter	**Jede Bewegung innerhalb der Software-Endschalter**	Eine Bewegung geht darüber hinaus

Prüfung	Grün	Gelb oder rot
Die sichere Höhe	**Sichere Höhe Z …** und wie weit über dem Rohteil und dem Spannmittel	Kein Platz über dem Rohteil, um hinüberzufahren
Die Tiefe	**Tiefe … unter der Oberseite** und, bei einem Rohteil im Schraubstock, wie weit es über die Backen hinausragt	Die Kugel ginge unter die Backen oder zu nah an die Unterseite des Rohteils

Folgt der Zwilling einer realen Maschine, sagt eine rote Zeile, dass Sie dort mit der eigenen Steuerung der Maschine antasten sollen.

Einen Zyklus zuerst im Zwilling ausprobieren

Die Prüfungen stammen aus einem Trockenlauf: Der Zwilling rechnet den Zyklus durch, ohne etwas zu bewegen. **Weg in 3D zeigen** zeichnet seine Bewegungen in der 3D-Ansicht anstelle des Programmwegs, bis Sie den Schritt verlassen.

Jede Berührung läuft gleich ab:

1. eine geschützte Bewegung zum Start (`G38.3`, F600);
2. eine schnelle Berührung (`G38.2`, F300);
3. zurückfahren;
4. eine langsame Berührung (F20): Das Ergebnis ist die Stelle, an der diese berührt hat.

Antastzyklen

Wählen Sie einen Zyklus, füllen Sie seine Felder aus und klicken Sie auf **Antasten starten…**. Jeder Zyklus beginnt 3 mm vor der Stelle, an der er die Fläche erwartet, und sucht 5 mm weiter. **Tiefe** ist, wie weit unter der Oberseite des Rohteils der Mittelpunkt der Kugel berührt (3 mm, solange Sie es nicht ändern). **Start**: **Vom Rohteil** arbeitet vom Rohteil in der Aufspannung aus; **X Y angeben** beginnt an einem Punkt, den Sie in Werkstückkoordinaten angeben.

Kante

Findet eine Fläche des Rohteils. **Welche Fläche**: **Links (X−)**, **Rechts (X+)**, **Vorne (Y−)** oder **Hinten (Y+)**. **Punkte**: **1 Punkt** oder **2 Punkte (auch die Drehung)**; das ergibt auch, wie weit das Rohteil verdreht ist.

Ergebnis: **Fläche X** oder **Fläche Y** und bei zwei Punkten **Drehung**.

Ecke

Findet eine Ecke aus zwei Flächen. **Welche Ecke**: **Vorne links**, **Vorne rechts**, **Hinten links** oder **Hinten rechts**; **Außen** ist die Ecke des Rohteils, **Innen** die Ecke einer Tasche (geben Sie ihre Lage mit **X Y angeben** an). **Abstand zur Ecke** gibt an, wie weit von der Ecke entfernt jede Fläche berührt wird (5 mm). **3 Punkte (auch die Drehung)** berührt die X-Fläche zweimal und ergibt auch die Drehung.

Ergebnis: **Ecke X**, **Ecke Y** und bei drei Punkten **Drehung**.

Bohrung und Zapfen

Bohrungsmitte tastet eine Bohrung von ihrer Mitte nach außen an; **Zapfenmitte** tastet einen Zapfen von außen an, über seine Oberseite hinweg. **Punkte**: **4 Punkte** oder **8 Punkte**. **Durchmesser** ist das Nennmaß; bleibt es leer, wird es von einem runden Rohteil übernommen. Ein Quader-Rohteil braucht den Durchmesser und **X Y angeben**.

Ergebnis: **Mitte X**, **Mitte Y** und **Durchmesser**.

Obere Fläche

Z-Oberfläche tastet die obere Fläche einmal an, in der Mitte des Rohteils oder bei **X Y angeben**.

Ergebnis: **Fläche Z**.

Den Werkstücknullpunkt aus dem Gefundenen setzen

Die Ergebniskarte zeigt jeden Wert in Maschinen- und Werkstückkoordinaten, jede Berührung und wie die Kalibrierung des Messtasters angewendet wurde. Ist das Rohteil verdreht, sagt sie, um wie viel; Sie können den Winkel der Aufspannung im Schritt **Aufspannung** darauf setzen.

So machen Sie das Ergebnis zum Werkstücknullpunkt:

1. Klicken Sie auf **Nullpunkt setzen…**
2. Haken Sie die Achsen an, die gesetzt werden sollen. Achsen, die der Zyklus nicht gemessen hat, lassen sich nicht anhaken: Eine Ecke ergibt kein Z. Ergebnis: Der Dialog zeigt den Satz, den er senden wird, zum Beispiel `G10 L2 P1 X-129.982 Y95.011`.
3. Klicken Sie auf **Als Nullpunkt G54 setzen** (oder die Nullpunktverschiebung, die in Gebrauch war).

Der Nullpunkt liegt auf dem, was gefunden wurde: der Fläche, der Ecke, der Mitte oder der Oberfläche. **Programm kopieren** kopiert die Sätze, die der Zyklus gesendet hat. **Erneut messen** führt zurück zu den Feldern.

> **Hinweis**
>
> Die Zahlen des Zwillings zeigen die Methode und den Ablauf. Sie sind nicht die Messgenauigkeit dieser Maschine.

Den Messtaster an einer Ringlehre kalibrieren

Ein Messtaster löst ein wenig nach der Berührung aus (Vorlaufweg), und nicht in jeder Richtung gleich. Das Kalibrieren an einer Ringlehre misst das, und spätere Zyklen werden damit korrigiert.

1. Stellen Sie die Ringlehre auf den Tisch und wählen Sie **Ringlehren-Kalibrierung**.
2. Geben Sie den **Ringdurchmesser** so ein, wie ihn das Zeugnis des Rings nennt.
3. **Position**: **Vom Rohteil (Bohrung des Rohrs)**, wenn die Ringlehre als Rohteil aufgespannt ist, oder **Angeben**: ihre Mitte.
4. Klicken Sie auf **Kalibrieren starten…** Ergebnis: **Wirksamer Kugelradius**, **XY-Vorlaufweg**, **Dreifach-Lobing** und **Ringmitte**.
5. Klicken Sie auf **Für diese Maschine speichern** oder auf **Nicht speichern**.

Die Kalibrierung gehört zur Maschine und zu diesem Kugeldurchmesser und ist Teil des Welt-Archivs der Maschine. Ein Zurücksetzen auf den Werkszustand vergisst sie. Jedes Ergebnis sagt, welche Kalibrierung es verwendet hat. Bis Sie kalibrieren, zeigt die Karte **Nicht kalibriert**: Eine Bohrung misst dann zwei Vorlaufwege zu groß und ein Zapfen zwei zu klein.

Die Maschine speichert eine Kalibrierung. Setzen Sie einen Messtaster mit einer anderen Kugel ein (etwa einen Taststift Ø0,5 für kleine Bohrungen), zeigt die Karte **Nicht kalibriert** und nennt Kugel und Ring, mit denen die Kalibrierung der Maschine gemacht wurde: Kalibrieren Sie die neue Kugel in der Ringlehre. Diese Kalibrierung ersetzt die vorige – zurück bei der ersten Kugel, kalibrieren Sie diese erneut.

Wenn ein Zyklus stoppt

Kann ein Zyklus nicht abgeschlossen werden, sagt die Ergebniskarte in Rot, warum, die Konsole protokolliert es, und **Erneut messen** führt Sie zurück zu den Feldern. Die üblichen Gründe:

- eine Berührung traf innerhalb ihres Suchwegs nichts: Die Fläche liegt nicht dort, wo der Zyklus sie erwartet hat. Die Steuerung löst ALARM:5 aus;
- der Messtaster hat auf dem Weg zu einer Berührung etwas berührt: Etwas steht dort, wo das Rohteil nicht sein sollte;
- die Berührungen ergeben keinen Kreis (Bohrung, Zapfen, Ring).

Folgen Sie nach einem Alarm den Schritten in der Maschinenleiste: **Reset**, dann **Entsperren** (siehe Alarme und was zu tun ist). Der Messtaster kann beim Stopp des Zyklus irgendwo stehen: Sehen Sie in die 3D-Ansicht, bevor Sie im Tippbetrieb fahren.

> **Achtung**
>
> Berührt der Messtaster, während die Maschine mit Vorschub halt angehalten ist, drücken Sie **Reset** und tasten Sie erneut an. **Zyklusstart** würde über den Kontakt hinausfahren.

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Programme erstellen (CAM)

Studio berechnet Werkzeugbahnen selbst. Sie beschreiben das Teil als Formelemente: eine Tasche, eine Kontur, Bohrungen, eine Gravur. Studio wählt die Schnittdaten mit den physikalischen Modellen des Zwillings und schreibt den G-Code. Danach führt es das Programm im Zwilling aus, bevor Sie fräsen.

Was das CAM tut

- Formen sind Daten. Die Form eines Formelements ist ein Rechteck, ein Kreis, ein Polygon, ein Pfad aus einer SVG- oder DXF-Datei, Text oder ein Netz (STL, 3MF, OBJ) für 3D-Arbeiten.
- Werkzeugbahnen werden berechnet. Studio berechnet den Versatz der Form, räumt die Fläche aus und verbindet die Bewegungen.
- Der Zwilling wählt die Schnittdaten. Drehzahl, Vorschub und Schnitttiefe kommen aus dem Werkstoff, dann aus den Modellen des Zwillings für die Leistung der Spindel, die Schnittkräfte, das Rattern und die Durchbiegung des Werkzeugs. Das sind Schätzwerte.
- Der Zwilling prüft das Programm. Studio kann das Programm im Zwilling ausführen, bevor Sie fräsen, und ändern, was der Lauf als zu viel zeigt.

Alles läuft auf Ihrem Computer. Zum Erstellen eines Programms ist keine Internetverbindung nötig.

Am schnellsten beschreiben Sie das Teil dem KI-Assistenten, zum Beispiel:

```
Make a program for a pocket 40 × 30 mm, 3 mm deep, in 6061-T6, with a 6 mm flat end mill.
```

Der Assistent verwendet das Tool `program_generate`. Er sagt, was er erstellt hat, wie lange es dauert und was der Zwilling gefunden hat.

Ein Auftrag: Formelemente, Werkzeuge und Werkstoff

Ein Auftrag ist das, was Studio über das Teil festhält:

- sein Name, der auch der Name des Programms ist, zum Beispiel `bracket.nc`;
- der Werkstoff, zum Beispiel 6061-T6 (Namen aus anderen Normen, etwa EN AW-6061, werden verstanden);
- das Werkzeug in der Spindel und die Werkzeugtabelle;
- die Formelemente, in der Reihenfolge, in der sie gefräst werden.

Jedes Programm, das Studio schreibt, wird zusammen mit dem Auftrag, der es erzeugt hat, als Version im Programmordner aufbewahrt. Beim erneuten Erstellen geht nichts verloren: Die älteren Versionen bleiben.

Der Assistent kann einen Auftrag öffnen und ändern, auch einen, den Sie im Arbeitsbereich CAM erstellt haben, zum Beispiel:

```
Zeigen Sie mir den Auftrag von bracket.nc, machen Sie dann seine Tasche 3 mm tief und entfernen Sie die zweite Bohrung.
```

Er verwendet das Tool `job_get`, um den Auftrag zu lesen, mit einer Zeile zu jeder Bearbeitung, und `job_edit`, um Werte zu ändern und Bearbeitungen zu entfernen, hinzuzufügen oder zu verschieben. Der geänderte Auftrag wird als nächste Version des Programms neu erstellt und wie zuvor im Zwilling geprüft. Lässt sich eine Änderung nicht ausführen, wird nichts geändert, und der Assistent sagt, warum. Nach den eigenen Versuchen des Zwillings (dem Regelkreis, der das Rattern beseitigt) liest der Assistent den Auftrag der letzten Version, mit dem, was diese Versuche geändert haben.

Bearbeitungen

Bearbeitung	Was sie fräst	Werkzeug
Planfräsen	Die Oberseite des Rohteils, eben	Schaftfräser
Tasche	Eine Fläche bis zu einer Tiefe: Rechteck, Kreis, Polygon oder eine Form	Schaftfräser
Kontur	Entlang einer Umrisslinie: außen, innen oder auf ihr	Schaftfräser
Nut	Eine Nut einer Breite entlang einer Linie	Schaftfräser
Bohren	Löcher an Punkten, bei Tiefe abgesetzt gebohrt	Bohrer
Ausbohren	Löcher, die größer sind als das Werkzeug, auf einer Schraubenlinie	Schaftfräser
Reiben	Gebohrte Löcher auf Maß	Reibahle
Gewinde	Innengewinde	Gewindefräser
Einstich	Eine Ringnut in der Bohrungswand: für einen Sicherungsring, einen O-Ring, eine Lichtfalle	T-Nutenfräser
Fase	Die Kante einer Umrisslinie	V-Fräser oder Fasenfräser
Gravieren	Text oder Linien	V-Fräser oder kleiner Schaftfräser
V-Gravur	Text oder Formen, die Tiefe folgt der Breite	V-Fräser
Perlage	Sich überlappende Kreise auf einer Fläche, die Verzierung der Uhrmacher	Schaftfräser so breit wie ein Kreis
3D-Schruppen, 3D-Schlichten	Ein Netz	Schaftfräser oder Kugelfräser

Im Arbeitsbereich CAM hat das Panel Formulare für Planfräsen, Taschen, Konturen und Fasen (eine importierte Form oder ein Rechteck, Kreis oder Vieleck, im Panel angegeben), Nuten, Bohren, Ausbohren, Gewindefräsen und Gravieren. Ein Auftrag mit anderen Bearbeitungen – von einem KI-Client oder aus einem Beispiel – öffnet sich trotzdem: Diese Bearbeitungen bleiben, wie der Auftrag sie angibt, und werden mit den übrigen erzeugt und geprüft. Ausgewählt, listet das Panel ihre Einstellungen unter den Namen von program_generate auf (Längen in mm); ändern lassen sie sich über einen KI-Client oder in der Datei des Auftrags.

Planfräsen, Taschen und Konturen

Eine Tasche wird in Schichten ausgeräumt. Ihre Strategie legt den Weg in jeder Schicht fest:

- Ringe (Standard): Ringe von der Mitte nach außen bis zur Wand;
- Zickzack: Bahnen hin und her;
- adaptiv: ein Weg, der den Eingriff des Werkzeugs gleichmäßig hält, für tiefe Taschen in härteren Werkstoffen.

Inseln in einer Tasche bleiben stehen. Die Restmaterialbearbeitung räumt nur aus, was ein größeres Werkzeug davor übrig gelassen hat.

Eine Kontur fräst entlang einer Umrisslinie, Schicht für Schicht, im Gleichlauf. Haltestege können ein Teil halten, das die Kontur herausfräst.

Nuten

Eine Nut wird in Schichten in voller Breite gefräst, dann bis auf ihre Breite erweitert. Eine Nut, die breiter ist als das Werkzeug, kann trochoidal gefräst werden: kleine Schleifen entlang ihrer Linie, für weniger Belastung des Werkzeugs.

Löcher: Bohren, Ausbohren, Reiben und Gewinde

- Das Bohren verwendet den Bohrzyklus der Steuerung. Ein Loch, das tiefer ist als das Dreifache des Bohrerdurchmessers, wird abgesetzt gebohrt.
- Das Ausbohren fräst ein Loch, das größer ist als der Schaftfräser, auf einer Schraubenlinie. Bis zum Doppelten des Werkzeugdurchmessers.
- Das Reiben bringt ein gebohrtes Loch mit einer Reibahle der Passung des Lochs auf Maß. Bohren Sie das Loch zuerst vor.
- Das Gewindefräsen schneidet ein Innengewinde mit einem Gewindefräser, von unten nach oben. Metrische (M8), Unified- (1/4-20 UNC), Rohr- (G1/4) und kegelige Rohrgewinde (1/4 NPT, Rc1/4) werden an ihren Namen erkannt.

Fasen, Gravieren und V-Gravur

Eine Fase folgt der Kante einer Umrisslinie mit einem V-Fräser, bis zu der Breite, die Sie angeben. Das Gravieren folgt Linien oder den Strichen eines Textes in einer Tiefe; Text beginnt an seinem X, wird darauf zentriert oder endet dort (**Ausrichtung** im Panel). Die V-Gravur fräst eine Form mit einem V-Fräser: Die Tiefe folgt der Breite der Form, sodass die Ecken scharf werden.

3D-Schruppen und -Schlichten

Ein Netz (STL, 3MF oder OBJ) wird in Ebenen geschruppt und dann geschlichtet. Die Schlichtstrategien sind parallel, Z-konstant, steil und flach sowie Bleistift.

Werkzeuge und die Werkzeugtabelle

Ein Werkzeug wird als seine Art, sein Durchmesser und das, was es sonst noch braucht, geschrieben, zum Beispiel:

Spezifikation	Werkzeug
`flat 6 n3 L20`	Schaftfräser, 6 mm, 3 Schneiden, 20 mm Auskraglänge
`ball 3 L12`	Kugelfräser, 3 mm
`drill 3.4 118 L20`	Bohrer, 3,4 mm, 118° Spitzenwinkel
`v 6 90 L18`	V-Fräser, 6 mm, 90°
`reamer 6 H7 n6 L32`	Maschinenreibahle, 6 mm, für H7-Löcher
`thread 5 60`	Gewindefräser, 5 mm, 60° Profil
`flat 2 n2 s4 l6 L10`	Schaftfräser, 2 mm, auf einem 4-mm-Schaft, der 6 mm über der Spitze beginnt
`thread 2 60 d1.4 n1 s4 l6 L9`	Gewindefräser, 2-mm-Kopf, 1,4-mm-Hals bis 6 mm, darüber ein 4-mm-Schaft

Geben Sie für jedes Werkzeug die Auskraglänge (`L`) an. Studio prüft dann, dass der Halter das Rohteil nicht erreicht, und der Z-Freiraum im Einrichteblatt kann das Werkzeug beurteilen.

Für einen kleinen Schaftfräser mit abgesetztem Schaft und für einen Hinterschnittfräser (Gewindefräser, T-Nutenfräser, Schwalbenschwanzfräser, Hinterschnitt-Kugelfräser) geben Sie auch den Schaftdurchmesser (`s`) an und wie hoch über der Spitze der Schaft beginnt (`l`). Wo der Schaft an den Wänden des Schnitts reiben würde, schneidet Studio nicht tiefer als `l`. Der Zwilling prüft den Schaft und den Körper des Messtasters auf Kollision mit Werkstück und Spannmittel.

Die Werkzeugtabelle listet die Werkzeuge nach ihren Nummern. Ein Formelement nennt das Werkzeug, das es verwendet, und Studio fügt die Werkzeugwechsel (`T2 M6`) hinzu.

Schnittdaten aus dem Zwilling

Studio geht von den Schnittdaten des Werkstoffs für das Werkzeug aus. Dann verwendet es die Modelle des Zwillings:

- die Leistung der Spindel: Der Schnitt nutzt höchstens etwa 80 % davon;
- das Rattern: Die Tiefe und die Drehzahl bleiben dort, wo das Stabilitätsmodell des Zwillings einen stabilen Schnitt erwartet;
- das Spannmittel: Die Kraft des Schnitts bleibt deutlich unter dem, was das Spannmittel hält;
- die Durchbiegung des Werkzeugs: Schlichtgänge werden leicht gehalten.

Bei eingeschalteter Verifikation führt Studio das Programm im Zwilling aus. Zeigt der Lauf Rattern oder eine zu hohe Last, ändert Studio das Programm, zum Beispiel mit flacheren Schichten, und versucht es erneut. Jede Version, die es ausprobiert, bleibt erhalten. Die Antwort sagt, was geändert wurde und warum.

Das sind Schätzwerte aus physikalischen Modellen. Verfolgen Sie den ersten Schnitt an der realen Maschine mit Augen und Ohren.

Passungen und Toleranzen

Schreiben Sie ein Maß mit seiner Passung, zum Beispiel `"d": "Ø12 H7"` für eine Bohrung oder `"fit": "g6"` für eine Welle. Studio fräst das Maß auf die Mitte des Toleranzfeldes der Passung und schlichtet die Wand in einem eigenen Durchgang.

Gängige Passungen lassen sich benennen: `sliding` (H7/g6), `locating` (H7/h6), `transition` (H7/k6), `press` (H7/p6).

Ein kleines, enges Loch wird besser gerieben als gefräst. Studio sagt das, und Sie können es bohren und dann mit einer Reibahle seiner Klasse reiben.

Der Zwilling schätzt den Rundlauffehler des Fräsers ab, und Studio berücksichtigt ihn bei den Maßen, die eine Passung haben.

Mehrere Aufspannungen

Ein Teil, das von zwei Seiten bearbeitet wird, braucht zwei Aufspannungen. Jede Aufspannung hat ihre eigenen Formelemente und sagt, wie das Teil zu ihr kommt: um X oder Y gewendet, anders gespannt, mit neu gefundenem Werkstücknullpunkt. Studio schreibt für jede Aufspannung ein Programm und prüft jedes im Zwilling an dem, was die vorherigen Aufspannungen hinterlassen haben. Wie Sie Aufspannungen hinzufügen und das Teil wenden: siehe Aufträge mit mehreren Aufspannungen.

Auf einer Platte verschachtelte Teile

Mehrere Teile lassen sich aus einer Platte fräsen. Studio platziert sie dicht beieinander, hält einen Rand ein und fügt Haltestege hinzu, damit die Teile bis zum Ende in der Platte bleiben.

Zoll-Maße und Brüche

Bei Zoll-Einheiten können Maße als Brüche geschrieben werden: `1/4`, `1 1/2`, `3/16"`. Zoll-Werkzeuge stehen in der Werkzeugbibliothek, zum Beispiel ein Schaftfräser von 1/4". Siehe Sprache und Einheiten.

Ein Programm im Zwilling prüfen

Führen Sie das Programm vor dem Fräsen im Zwilling aus. Der Zwilling fräst sein eigenes Werkstück und meldet Kollisionen, Überschnitte, Rattern, die Last der Spindel und die Zeit. Siehe Simulation.

> **Achtung**
>
> Der Zwilling prüft das Programm gegen die Aufspannung, die Sie ihm gegeben haben. Richten Sie den Zwilling so ein, wie die reale Maschine ist: dasselbe Rohteil, dasselbe Spannmittel, dieselben Werkzeuge und derselbe Werkstücknullpunkt.

Das Einrichteblatt

Ein Einrichteblatt sagt dem Bediener, wie der Auftrag einzurichten ist: das Rohteil, das Spannmittel, die Werkzeuge, der Werkstücknullpunkt und der Z-Freiraum. Siehe Einrichteblätter.

Programme und der Editor

Der Arbeitsbereich **Programm** (Alt+3) enthält Ihre Programme und die Beispielteile, einen Editor, der ein Programm während der Eingabe prüft, und eine Vorschau darauf, was das Programm mit dem Werkstück macht. Studio legt jedes Mal, wenn Sie ein Programm speichern oder in die Maschine laden, eine Version davon an, sodass Sie sehen können, was sich geändert hat, und zurückgehen können.

Ein Programm öffnen

Der Arbeitsbereich hat drei Teile:

- links **Programme** (jedes Programm und seine Versionen) und **Beispielteile**;
- in der Mitte der Editor;
- rechts die 3D-Vorschau, **Laden und prüfen** und die Befunde der Programmprüfung.

So öffnen Sie ein Programm:

- **Datei > Programm öffnen…** (Strg+O) lädt eine G-Code-Datei in die Maschine und, wenn der Editor keine ungespeicherten Änderungen hat, auch in den Editor. Studio liest Ihre Datei und schreibt nie in sie.
- Eine G-Code-Datei (.nc, .gcode, .ngc, .tap, .txt, .cnc) irgendwo auf das Fenster zu ziehen, bewirkt dasselbe. Während ein Programm läuft, öffnet Studio nichts und meldet das in der Konsole.
- **Datei > Zuletzt verwendete Programme** listet die letzten acht aus einer Datei geöffneten Programme auf, das neueste zuerst. Eines, das inzwischen verschoben oder gelöscht wurde, verschwindet aus der Liste, wenn Sie es wählen; **Liste leeren** leert sie.
- **Datei > Programm drucken…** (Strg+P) druckt das Programm in der Maschine, mit nummerierten Zeilen, schwarz auf weiß, oben auf jeder Seite mit seinem Namen, der Seitenzahl und dem Datum.
- Doppelklicken Sie unter **Programme** auf ein Programm oder eine Version: Der Text kommt in den Editor. In die Maschine gelangt er erst, wenn Sie auf **Laden und prüfen** klicken.

Der Editor und die Maschine können verschiedene Programme enthalten. Hat die Maschine ein Programm, das Sie nicht im Editor geöffnet haben, sagt die Statuszeile das: Strg+K lädt das Programm des Editors in die Maschine, Strg+Umschalt+R übernimmt das Programm der Maschine in den Editor (Ihre Änderungen werden zuerst als Version gespeichert).

Beispielprogramme

Studio 1.0 enthält vier Beispielteile: eine Motorplatte für NEMA 17, eine Frontplatte für ein Messgerät, einen Mikrofluidik-Chip für Tröpfchen und ein Isogrid-Leichtbaupaneel. Weitere kommen mit Updates. Doppelklicken Sie unter **Beispielteile** auf eines oder wählen Sie es über **Datei > Beispielteile…**.

Das Laden eines Beispiels richtet die ganze Maschine dafür ein: das Rohteil und das Spannmittel, die Werkzeugtabelle und das Werkzeug in der Spindel, die Referenzfahrt, den Werkstücknullpunkt, das Programm und das Bauteil zum Vergleich, wenn das Beispiel eines hat.

> **Hinweis**
>
> Ein Beispiel ersetzt das Rohteil, die Werkzeugtabelle und den Werkstücknullpunkt G54, die Sie bisher hatten. **Demoprogramm** in derselben Liste lädt nur das Programm und lässt das Rohteil, wie es ist.

Der Editor

Der Editor ist ein G-Code-Editor:

- Die Wörter sind nach Art eingefärbt: G- und M-Codes, Achsen, Kreisbogenwörter, Vorschub, Drehzahl, Werkzeug, Kommentare, Ausdrücke.
- Der Rand zeigt die Zeilennummern und Markierungen: die Befunde der Prüfung (rote Fehler, gelbe Warnungen, blaue Hinweise), Haltepunkte und den Satz, den die Maschine gerade ausführt.
- Nach einem Lauf zeigt ein zweiter Rand die Zeit jedes Satzes in Sekunden: gelb, wenn der Vorschub nicht erreicht wurde, rot, wenn dort etwas geschehen ist (ein Eilgang ins Material, ein Schnitt bei stehender Spindel, eine Kollision, ein Alarm). Klicken Sie darauf, um die Werte des Satzes aus dem letzten Lauf zu sehen.

Die Statuszeilen unter dem Editor zeigen Zeile und Spalte, ob der Text in der Maschine ist, die Version, die Anzahl der Befunde der Prüfung und den Zustand des Programms nach dem Satz, in dem Sie sich befinden: die modalen Codes, den Vorschub, die Drehzahl, das Werkzeug und die Position.

Vergrößern gibt dem Editor den ganzen Arbeitsbereich; **Wiederherstellen** bringt die Programme und die Vorschau zurück.

Farben und Zeilennummern

Die Farben folgen dem Erscheinungsbild des Fensters (hell oder dunkel). Sätze, die die Steuerung über den gerade ausgeführten Satz hinaus schon im Voraus übernommen hat, sind grau.

Suchen und Ersetzen

- Strg+F öffnet das Suchfeld. Die Eingabetaste sucht den nächsten Treffer, Umschalt+Eingabetaste den vorherigen. Die Suche unterscheidet nicht zwischen Groß- und Kleinschreibung und springt am Ende wieder an den Anfang.
- Strg+G springt zu einer Zeilennummer.

- Strg+H öffnet das Ersetzen-Feld unter dem Suchfeld. Die Eingabetaste ersetzt den ausgewählten Treffer und springt zum nächsten; Strg+Eingabetaste ersetzt alle Treffer. Strg+Z macht es rückgängig.

Esc schließt die Felder.

Haltepunkte und Pausen

Ein Haltepunkt hält die Zeit des Zwillings an, wenn der Lauf diesen Satz erreicht; die Bewegung der Maschine ändert sich nicht. F5 setzt fort.

- Strg+B oder ein Klick in den Markierungsrand setzt einen Haltepunkt auf den Satz.
- **Anhalten bei** (Strg+Umschalt+B) hält den Zwilling an, wenn etwas geschieht: Das Werkzeug beginnt zu schneiden, ein Eilgang schneidet Material, eine Beinahekollision oder eine Kollision, Rattern, ein Alarm, eine Änderung des Zustands der Steuerung, die Schnittkraft, die Last der Spindel oder der Schleppfehler über einer von Ihnen gewählten Grenze oder ein Wechsel bei den modalen Codes. Läuft der Zwilling in Echtzeit, wird die Ratterprüfung neben der Simulation berechnet; ein Anhalten bei Rattern und die Ratterwerte in den Ansichten können daher etwas nach dem Schnitt kommen: meist einige hundert Millisekunden, eine oder zwei Sekunden dort, wo viele neue Schnittbedingungen zusammenkommen (eine Tasche um Inseln). Bis die Prüfung für ein Werkzeug geantwortet hat, zeigt der Ratterkennwert **wird berechnet…**; die Kurven und die Werkzeugspur warten auf die Antwort zu jedem Schnitt und zeigen so den Kennwert dieses Schnitts. Der Bericht am Ende des Laufs ist in beiden Fällen derselbe.
- Das Menü des Satzes (Rechtsklick auf den Rand oder Umschalt+F10) enthält außerdem **Bis Satz … ausführen** und **Alle Haltepunkte und Bedingungen löschen**.

Haltepunkte wirken, sobald das Programm in der Maschine ist.

Die Programmprüfung

Der Editor prüft das Programm während der Eingabe (einen Moment, nachdem Sie aufgehört haben), beim Laden und noch einmal nach einem Lauf. Die Prüfung hat zwei Teile:

- Die Regeln prüfen jeden Satz darauf, was die Steuerung und die Maschine akzeptieren. Sie brauchen nichts weiter.
- Die Vorschauprüfung lässt das Programm gegen das Werkstück laufen, wie es jetzt ist. Sie braucht eine eingeschaltete und referenzierte Maschine und kein laufendes Programm. Beim Tippen nimmt sie Programme von höchstens 20.000 Sätzen; ein längeres Programm wird ganz geprüft, sobald es in die Maschine geladen ist, und seine Probleme erscheinen als rote Kästen in der 3D-Ansicht und im Reiter **Meldungen**.

Programm > Programm prüfen ist eine weitere Prüfung: Sie schickt das geladene Programm Satz für Satz durch den eigenen Prüfmodus der Steuerung, während sich die Maschine nicht bewegt.

Was die Prüfung findet

Jeder Befund ist ein Fehler, eine Warnung oder ein Hinweis und wird unter seinem Satz mit der Nummer seiner Regel angezeigt. Beispiele:

Regel	Art	Was sie findet
S01	Warnung	Kein Startabschnitt vor der ersten Bewegung: Das Programm verlässt sich auf den Zustand der Steuerung
S02	Warnung	Kein `M2` oder `M30` am Ende
S04	Fehler	Eine Schnittbewegung, bevor ein `F` gesetzt wurde
S05	Warnung	Die Spindel startet ohne `S`, oder `S` liegt außerhalb des Bereichs der Spindel
S07	Fehler	Ein Code, den die Steuerung nicht annimmt: Ausdrücke, `O`-Wörter, Unterprogramme, `G41`/`G42`, `G68`
S10	Warnung	Ein Eilgang unterhalb der Oberkante des Werkstücks und tiefer, als das Programm geschnitten hat
S11	Fehler	Ein Kreisbogen, den die Steuerung ablehnt
S12	Warnung	Die wassergekühlte Spindel schneidet ohne ihr Kühlmittel (`M8`)
S14	Warnung	Ein Werkzeug mit anderer Länge als das, mit dem der Werkstücknullpunkt gesetzt wurde, und nichts misst es

Die Vorschauprüfung ergänzt, was das Programm mit dem Werkstück machen würde: Eilgänge ins Material, Schneiden bei stehender Spindel, Kollisionen mit dem Spannmittel, Bewegungen über den Verfahrbereich hinaus und eine Spindel, die trocken läuft.

Einen Befund beheben

Manche Befunde kommen mit einem Vorschlag, zum Beispiel **Vorschlag (davor einfügen): M8**.

1. Setzen Sie den Cursor in den Satz, oder gehen Sie mit Alt+↓ und Alt+↑ von Befund zu Befund.
2. Drücken Sie Strg+. (oder wählen Sie **Vorschlag anwenden** im Menü des Satzes).

Ergebnis: Die Änderung erfolgt in einem Schritt; Strg+Z macht sie rückgängig. Lesen Sie den Vorschlag, bevor Sie ihn anwenden: Er behebt den Befund, nicht Ihre Absicht.

Optimieren

Optimieren (Strg+Umschalt+O) sucht ein schnelleres Programm, das dasselbe Bauteil fertigt: Es ändert Vorschübe und Drehzahlen, fasst kurze Sätze zusammen und senkt die Rückzüge. Es lässt das ursprüngliche und das optimierte Programm auf zwei verborgenen Kopien der Maschine laufen, mit derselben Aufspannung. Es braucht ein eingerichtetes Rohteil. **Optimieren** gibt es ab der Edition Pro.

Ist das Ergebnis besser, ersetzt es den Text des Editors (Strg+Z bringt das Original zurück), und der Vergleich öffnet sich. Die Konsole nennt die Zeiten, die höchste Schnittkraft und den Ratterkennwert vorher und nachher sowie was sich geändert hat. Klicken Sie auf **Optimierung stoppen** oder drücken Sie Strg+Umschalt+O, um die Optimierung zu beenden.

Versionen

Studio speichert eine Version eines Programms, wenn Sie:

- auf **Version speichern** klicken (Strg+S);
- es in die Maschine laden (**Laden und prüfen** oder **Datei > Programm öffnen…**);
- es optimieren oder ein erzeugtes Programm oder das Programm der Maschine anstelle Ihrer Änderungen übernehmen (Ihre Änderungen werden zuerst gespeichert).

Eine Version, die der letzten gleicht, wird nicht noch einmal gespeichert. Versionen liegen in `Dokumente\Twinwright Studio\programs\<program>\`, je eine Datei, und werden nie ausgedünnt. **Programme** listet sie unter jedem Programm auf, mit der Zeit und dem Grund, aus dem jede gespeichert wurde.

Zwei Versionen vergleichen

Vergleichen (Strg+D) zeigt unter dem Editor die neueste gespeicherte Version, die sich vom Text des Editors unterscheidet: geänderte Zeilen gelb, hinzugefügte grün, entfernte rot und Zeilen, in denen sich nur `F` oder `S` geändert hat, blau. Drücken Sie Strg+D noch einmal für eine ältere Version, Alt+↓ und Alt+↑, um von Unterschied zu Unterschied zu gehen, und Esc zum Schließen.

Die Überschrift zählt die Unterschiede und vergleicht, wenn beide Versionen in dieser Sitzung bis zum Ende gelaufen sind, ihre Laufzeiten.

Um zwei gespeicherte Versionen zu vergleichen, öffnen Sie zuerst die neuere im Editor und drücken Sie dann Strg+D.

Zu einer Version zurückgehen

1. Doppelklicken Sie unter **Programme** auf die Version. Ergebnis: Der Text der Version kommt in den Editor. Ihre ungespeicherten Änderungen werden zuerst als Version gespeichert; Strg+Z macht die Änderung rückgängig.
2. Klicken Sie auf **Laden und prüfen** (Strg+K), um sie in die Maschine zu laden.

Speichern und Exportieren

- **Version speichern** (Strg+S) legt eine Version im Programmordner von Studio ab.
- Strg+Umschalt+S speichert das Programm des Editors als Datei an einem Ort Ihrer Wahl.
- Eine Version ist eine einfache `.nc`-Datei: Sie können sie aus dem Programmordner kopieren.

Datei > Kurven exportieren (CSV)… speichert die Kurven des Laufs (siehe Analyse und Diagnose). Das Einrichteblatt und der Prüfbericht befinden sich auf der Karte **Berichte** in **Vorbereiten** (siehe Prüfung).

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Ein Programm ausführen und fortsetzen

Der Arbeitsbereich **Ausführen** führt das geladene Programm aus und zeigt, was die Maschine tut, während sie schneidet. Wenn ein Lauf stoppt, nennt Studio den nächsten Schritt und kann das Programm ab einem Satz erneut starten.

Laden und Starten

Um ein Programm in die Maschine zu laden, gehen Sie auf eine dieser Arten vor:

- **Datei > Programm öffnen…** (Strg+O): Wählen Sie eine G-Code-Datei. Studio liest sie; es schreibt nie in Ihre Datei.
- **Datei > Beispielteile…**: Wählen Sie ein Beispiel. Studio richtet dafür auch das Rohteil, die Werkzeuge und den Werkstücknullpunkt ein.
- Im Arbeitsbereich **Programm** lädt **Laden und prüfen** (Strg+K) das Programm im Editor.

Die Maschinenleiste zeigt dann den Namen des Programms, die Anzahl seiner Sätze und etwa, wie lange es dauert.

Zum Starten klicken Sie in der Maschinenleiste auf **Zyklusstart** (F5). **Zyklusstart** ist farbig gefüllt, wenn das Programm geladen ist und die Maschine im Leerlauf und referenziert ist. Hat die Programmprüfung Probleme gefunden, sagt die Konsole das beim Start des Programms.

Bevor Sie ein Programm auf einer echten Maschine ausführen, prüfen Sie es: **Programm > Programm prüfen** schickt es durch den Prüfmodus der Steuerung. Die Steuerung liest jeden Satz, und die Maschine bewegt sich nicht.

Einen Lauf beobachten

Während ein Programm läuft:

- Die 3D-Ansicht zeigt das Werkzeug, die Spur seiner Spitze und das Werkstück, wie es geschnitten wird. **Werkzeug-Nahansicht** zeigt das Werkzeug im Eingriff.
- Der Reiter **Programm** im Dock zeigt die Programmliste: Der ausgeführte Satz ist markiert, erledigte Sätze sind heller, und ein Balken zeigt die Zeit jedes Satzes. **Aktuellem Satz folgen** hält den Satz im Blick.
- **Programmstruktur** neben der Programmliste teilt das Programm in seine Abschnitte und Ebenen (aus den Kommentaren des Programms) mit ihren Zeiten auf.
- Die rechte Spalte zeigt die Anzeigen.

Wenn das Programm endet, sagt der Reiter **Meldungen**, wie viele Sätze gelaufen sind und wie lange der Lauf in der Simulation gedauert hat.

Fortschritt und verbleibende Zeit

Die Maschinenleiste zeigt den ausgeführten Satz von allen Sätzen, den Anteil der erledigten Sätze sowie **Lauf** und **Rest**:

- **Lauf** ist die bisherige Maschinenzeit, in der Zeit der Simulation;
- **Rest** ist die Schätzung für die verbleibende Zeit, aus den Vorschüben und Eilgängen des Programms.

Nach einem Alarm zeigt die Leiste den Satz, bei dem das Programm angehalten hat.

Die Anzeigen

Die rechte Spalte von **Ausführen**, von oben:

- **Position**: jede Achse in Werkstück- und Maschinenkoordinaten, mit einer Markierung, wenn die Achse referenziert ist. Darunter die Zeile der modalen Codes: die geltenden G-Codes, das Werkzeug, der Vorschub, die Spindeldrehzahl und die M-Codes. Bei einem Alarm zeigen die Spindel und das Kühlmittel den Zustand, den die Maschine hat, und die Werte des Programms stehen im Tooltip;
- **Spindel und Vorschub**: die Spindeldrehzahl (der Bogen ist die Drehzahl, die Marke die Solldrehzahl), die Last der Spindel (grün bis 80 %, rot über 100 %) und der Vorschub;
- **Schnitt** (ab Stufe Ingenieur): die Schätzungen des Zwillings für die Schnittkraft, die Leistung, den Ratterkennwert und den Vorschub pro Zahn;
- **Overrides**, **Bedienfeld**, **Handbetrieb** und **Werkzeuge und Nullpunkt** (ab Stufe Bediener).

Klicken Sie auf den Titel eines Abschnitts, um ihn einzuklappen. Im Werkstattmodus zeigt **Position** zusätzlich **Restweg**: den Weg, der im aktuellen Satz noch bleibt, für jede Achse.

Vorschub halt, Stopp und Overrides

- **Vorschub halt** (F6 oder Leertaste) bremst die Achsen bis zum Stillstand ab; die Spindel dreht weiter. **Zyklusstart** setzt das Programm fort.
- **Stopp** (F7) hält zuerst an und setzt die Steuerung zurück, sobald die Maschine steht. Drücken Sie ihn erneut, um sofort zurückzusetzen.
- **Reset** (F12) stoppt das Programm sofort.

Die Overrides ändern den Vorschub und die Spindeldrehzahl des Programms, während es läuft, von 10 % bis 200 %:

Override	Tasten	Auf dem Bildschirm
Vorschub	Strg+↑, Strg+↓: ±10 %	**Overrides**: Regler ziehen; mit Strg in Schritten von 1 %
Drehzahl	Strg+→, Strg+←: ±10 %	**Overrides**: Regler ziehen
Eilgang	**Maschine > Overrides**: 25 %, 50 %, 100 %	**Overrides**: 100, 50, 25

Strg+0 oder ein Doppelklick auf einen Regler setzt den Vorschub und die Drehzahl auf 100 % zurück.

Einzelsatz

Einzelsatz in der Maschinenleiste hält das Programm nach jedem Satz an. Klicken Sie auf **Zyklusstart** oder wählen Sie **Programm > Nächster Satz**, um den nächsten Satz auszuführen. Sie können ihn vor dem Start oder während des Laufs aktivieren; klicken Sie erneut darauf, um ohne Halt weiterzulaufen.

Tippbetrieb und Handrad

Den Tippbetrieb nutzen Sie in **Ausführen** unter **Handbetrieb** (der Abschnitt ist zunächst eingeklappt) oder mit den Tasten (siehe Die Tasten für die Maschine).

- **Schrittweite**: 0,01; 0,1; 1; 10 oder 50 mm pro Tastendruck oder **Gedrückt halten**: Die Achse fährt, solange Sie die Taste halten.
- **Geschwindigkeit**: 100, 300, 1500 oder 3000 mm/min.
- Das Tastenfeld: X und Y im Kreuz, Z daneben, **Tippen stoppen** (Esc) in der Mitte.
- **Handrad**: Wählen Sie **X**, **Y** oder **Z** und das Inkrement (**×1**, **×10**, **×100**: 0,001; 0,01 oder 0,1 mm pro Raste), und ziehen Sie dann die Drehscheibe oder drehen Sie das Mausrad darüber.

Der Tippbetrieb ist nur möglich, wenn die Maschine eingeschaltet ist, im Leerlauf steht, keinen Alarm hat und kein Programm läuft. Eine gehaltene Tippbewegung stoppt, wenn Sie die Taste loslassen oder das Fenster den Fokus verliert.

Extras > Einstellungen… > Bedienfeld und Geräte > Handrad legt fest, wie sich das Handrad verhält: **Geschwindigkeit** stoppt die Achse, wenn das Rad stehen bleibt; **Weg** fährt jede Raste.

Alarme und was zu tun ist

Bei einem Alarm wird der Zustand in der Maschinenleiste rot, und die Leiste zeigt:

- den Alarm und was er bedeutet, zum Beispiel **ALARM:17 Ein Motorantrieb meldet einen Fehler**;
- **Nächster Schritt:** was jetzt zu tun ist, mit der Taste, die es ausführt;
- **Alle Schritte**, wenn es mehr als einen gibt.

Klicken Sie auf die Taste oder verwenden Sie **Maschine > Nächsten Schritt ausführen**, um den nächsten Schritt auszuführen. Nach jedem Schritt ermittelt Studio die Schritte neu aus dem Zustand der Maschine.

Die weiteren Anzeichen eines Alarms: eine Zeile unter **Position**, wenn man der Position nicht mehr trauen kann, der rote Satz in der Programmliste, an dem das Programm angehalten hat, und eine Beschriftung am Antrieb in der 3D-Ansicht.

Die Alarme, ihre Ursachen und was sie beseitigt: Alarme.

Die Schritte zurück in den Betrieb

Alle Schritte öffnet die Schritte zurück in den Betrieb: die Ursache, dann die Schritte der Reihe nach, jeden mit seiner Taste und dem Grund, warum er nötig ist. Der Schritt, der jetzt dran ist, ist markiert. Ein Klick führt nur den nächsten Schritt aus.

Zum Beispiel nach einem Antriebsalarm (ALARM:17):

1. **Not-Halt drücken, Antriebe stromlos schalten** (F1): Der Alarm eines Antriebs erlischt nur, wenn der Antrieb stromlos wird.
2. **Not-Halt entriegeln** (F1).
3. **Einschalten** (F2): Die Antriebe haben wieder Spannung.
4. **Reset** (F12): Die Steuerung behält ALARM:17, nachdem sich die Antriebe erholt haben.
5. **Referenzfahrt** (F3): Die Spannung war aus, daher ist der Position nicht zu trauen.

Wurde der Fehler in **Diagnose** injiziert, sagt die Ursache das, mit der Uhrzeit.

Hat ein Alarm ein Programm gestoppt, enden die Schritte mit dem Satz, ab dem neu gestartet wird, zum Beispiel dem Anfang der Ebene, in der es gestoppt hat.

Ab einem Satz neu starten

Studio kann ein Programm ab jedem Satz starten. Es schreibt einen Startabschnitt, der die Maschine so einrichtet, wie das Programm sie an diesem Satz hätte, und führt dann den Rest aus.

1. Stoppen Sie das Programm. Die Maschine muss eingeschaltet sein, und es darf kein Programm laufen.
2. Klicken Sie im Reiter **Programm** des Docks mit der rechten Maustaste auf den Satz und wählen Sie **Ab Satz … starten**, dann **OK**. Im Editor klicken Sie mit der rechten Maustaste auf den Rand des Satzes oder drücken Umschalt+F10. Ergebnis: Studio lädt „*Name* (ab Satz *n*)“, startet es aber nicht.
3. Klicken Sie auf **Zyklusstart**.

Der Startabschnitt führt folgende Schritte aus:

1. Z ganz nach oben fahren;
2. die Einheiten, die Ebene, den Distanzmodus und die Nullpunktverschiebung setzen, die das Programm an diesem Satz hatte;
3. das Werkzeug wechseln, wenn das Programm es vor diesem Satz gewechselt hatte;
4. das Kühlmittel und die Spindel mit der Drehzahl des Programms einschalten und warten, bis die Spindel ihre Drehzahl erreicht hat;
5. in sicherer Höhe über den Startpunkt fahren, bis 2 mm darüber absenken und dann mit Vorschub nach unten fahren;
6. das Programm ab diesem Satz fortsetzen.

Die Vorschauprüfung prüft dann den Rest gegen das Werkstück, wie es jetzt ist.

> **Achtung**
>
> Studio kann manche Sätze vor dem Start nicht nachvollziehen: Parameter, Ausdrücke, Unterprogramme, `G10`, `G28`, `G30` und `G92`. Die Hinweise beim Laden listen sie auf. Prüfen Sie den Zustand der Maschine, bevor Sie **Zyklusstart** drücken.

Den Start zuerst an einer Kopie ausprobieren

Nachdem ein Alarm ein Programm gestoppt hat, enden die Schritte mit **Vorschau Startabschnitt…**. Das öffnet den Startabschnitt und seine Hinweise.

1. Klicken Sie auf **Auf einer Kopie testen**. Ergebnis: Der Start läuft auf einer verborgenen Kopie der Maschine, ausgehend vom Werkstück, wie es jetzt ist, und Studio meldet seine Zeit, jede Kollision, den höchsten Ratterkennwert und wie viel Material er abgetragen hat.
2. Wenn es richtig aussieht, folgen Sie den Schritten zurück in den Betrieb und starten Sie dann ab dem Satz.

Ausführen mit der Simulationsgeschwindigkeit

Der Zwilling läuft in Echtzeit. **Simulation > Geschwindigkeit** lässt ihn mit ×0,5 bis ×50 der Echtzeit laufen. Die Uhr in der Maschinenleiste zeigt die Zeit der Simulation, die gewählte Geschwindigkeit und die Geschwindigkeit, die der Computer erreicht.

Simulation > Simulation pausieren oder die Sanduhr in der Maschinenleiste hält die Zeit des Zwillings an: Solange sie steht, ändert sich nichts. **Zyklusstart** setzt fort. Haltepunkte und Pausenbedingungen im Editor halten die Zeit des Zwillings auf dieselbe Weise an (siehe Programme und der Editor).

Das Pausieren der Simulation ist kein Vorschub halt: Der Zustand der Maschine ist eingefroren, nicht angehalten.

Analyse und Diagnose

Analyse blickt auf einen Lauf zurück: wohin die Zeit gegangen ist, die Schätzungen des Zwillings für Kraft und Rattern und wie das gefräste Werkstück im Vergleich zur Konstruktion abschneidet. **Diagnose** zeigt die Leistungskette und die Signale der Maschine und lässt Sie Fehler in den Zwilling injizieren, um das Auffinden zu üben.

Einen Lauf ansehen

Analyse (Alt+5, ab Stufe Ingenieur) füllt sich, sobald ein Programm gelaufen ist.

- **Läufe** links listet die Läufe dieser Sitzung auf: das Programm, wie der Lauf endete, seine Maschinenzeit und wann er gestartet wurde. Läufe, die mit einem Alarm oder einem Fehler endeten, sind rot.
- In der Mitte stehen die Kurven und darunter zwei Reiter: **Satzprofil** und **Stabilitätskarte**.
- **Letzter Lauf** rechts fasst den letzten Lauf zusammen: wie weit er kam, die Sätze, die am meisten Zeit brauchten, die höchste Schnittkraft und den höchsten Ratterkennwert mit ihren Sätzen, das abgetragene Material, die Sätze, in denen der Vorschub nicht erreicht wurde, und die Ereignisse.

Die Kurven, das Profil und die Zusammenfassung gehören zum letzten Lauf.

Satzprofil listet die Sätze auf, die Zeit gebraucht haben, den längsten zuerst: die Zeit und ihren Anteil, den Soll-Vorschub und den erreichten Vorschub, warum er nicht erreicht wurde, die Schnittkraft, den Ratterkennwert, das abgetragene Material und die Ereignisse. Doppelklicken Sie auf einen Satz oder drücken Sie die Eingabetaste darauf, um im Editor **Programm** zu ihm zu springen.

Warum ein Satz seinen Vorschub nicht erreicht hat:

- die Höchstgeschwindigkeit der Achsen;
- Ecken, die zu dicht beieinander liegen, um dazwischen zu beschleunigen;
- die Sehnengrenze des Kreisbogens;
- die Planung der Steuerung; das Sender-Programm oder das Einlesen der Steuerung kommt nicht hinterher;
- ein Vorschub-Override unter 100 % oder ein Vorschub halt.

Zeit: geschätzt und tatsächlich

Vor einem Lauf schätzt Studio die Maschinenzeit aus den Vorschüben und Eilgängen des Programms: Die Maschinenleiste zeigt diese Zeit mit **etwa**, und die **Programmstruktur** im Dock zeigt sie für jeden Abschnitt. Während und nach einem Lauf sind die Zeiten die Maschinenzeit des Zwillings, mit der Beschleunigung und der Planung der Steuerung darin: Die Maschinenleiste zeigt die gelaufene und die verbleibende Zeit, die **Programmstruktur** die Zeit des ganzen Laufs und das **Satzprofil** die Zeit jedes Satzes.

Alle Zeiten sind Maschinenzeit in der Simulation, nicht die Zeit Ihrer Uhr.

Kraft, Rattern und Spanungsdicke

Die Kurven zeigen die Werte des Laufs über die letzten 10, 30 oder 120 s: den Vorschub, die Ist-Geschwindigkeit, die Drehzahl und die Last der Spindel, die Schnittkraft, das Drehmoment und die Leistung, die Spanungsdicke, den Ratterkennwert und die Schleppfehler. Klicken Sie auf den Namen einer Kurve, um sie ein- oder auszublenden.

Der Ratterkennwert ist die Schnitttiefe geteilt durch die Tiefe, die bei der verwendeten Spindeldrehzahl stabil ist:

Kennwert	Bedeutung
Unter 0,8	Stabil
0,8–1	Nahe an der Grenze
Über 1	Rattert

Stabilitätskarte zeichnet die stabile Schnitttiefe über der Spindeldrehzahl, für den letzten Schnitt (oder, vor dem ersten Schnitt, für eine volle Nut von 1 mm Tiefe entlang X). Klicken Sie auf **Stabilitätskarte berechnen**. Ein roter Punkt markiert den Schnitt, wie er war, eine grüne gestrichelte Linie die stabilste Drehzahl.

Ansicht > Einfärbung färbt die Spur der Werkzeugspitze nach Vorschubgeschwindigkeit, Schnittkraft, Spanungsdicke, Ratterkennwert oder Programmsatz und das Werkstück nach Material und Tiefe, Abweichung vom Bauteil oder Programmsatz.

Dies sind die Schätzungen des Zwillings aus einem physikalischen Modell (siehe Zwillingsmodelle).

Mit dem Bauteil aus der Konstruktion vergleichen

Um zu sehen, wo das gefräste Bauteil von der Konstruktion abweicht:

1. Klicken Sie in **Vorbereiten** im Schritt **Aufspannung** auf **Bauteil-STL (Vergleich)…** und wählen Sie die STL-Datei des Bauteils (in mm, ihr Ursprung im Nullpunkt des Programms).
2. Führen Sie das Programm aus.
3. Wählen Sie **Ansicht > Einfärbung > Werkstück: Abweichung vom Bauteil**.

Ergebnis: Das Werkstück wird danach eingefärbt, wie weit jeder Punkt vom Bauteil entfernt ist: Innerhalb von ±20 µm gilt als gut, die anderen Farben zeigen stehen gebliebenes Material und Überschnitte. **Vergleich beenden** entfernt das Bauteil. Die Beispielteile laden ihr eigenes Bauteil.

Zwei Läufe vergleichen

Analyse zeigt den letzten Lauf. Um zwei Läufe eines Programms zu vergleichen:

- Im Editor **Programm** gibt **Vergleichen** (Strg+D) die Laufzeiten beider Versionen und die Zeit der geänderten Sätze an, wenn beide in dieser Sitzung bis zum Ende gelaufen sind (siehe Zwei Versionen vergleichen).
- Der KI-Assistent kann zwei Läufe nebeneinander darstellen (siehe Der KI-Assistent).

Exportieren

Kurven exportieren (CSV)… (in **Letzter Lauf** oder **Datei > Kurven exportieren (CSV)…**) speichert etwa die letzten 160 s der Kurven als CSV-Datei: eine Zeitspalte in Sekunden, dann eine Spalte für jede Kurve, mit ihrer Einheit in der Überschrift und den Werten in den Anzeigeeinheiten.

Werkstück exportieren… in **Vorbereiten** oder **Datei > Werkstück exportieren…** speichert das gefräste Werkstück als STL-Datei (in Programm-, Rohteil- oder Maschinenkoordinaten) oder als Höhenkarte.

Der Arbeitsbereich Diagnose

Diagnose (Alt+6, ab Stufe Techniker) zeigt die Leistungskette der Maschine als Diagramm: Die Leistung fließt von links nach rechts. Durchgezogene Linien führen Leistung, grün, wenn sie unter Spannung stehen; gestrichelte Linien sind Signale. Jede Komponente ist grün, wenn sie in Ordnung ist, grau, wenn sie aus ist, gelb bei einer Warnung und rot bei einem Alarm oder einem Fehler.

Klicken Sie auf eine Komponente, um ihre Details in der Spalte rechts zu sehen. Unter dem Diagramm stehen die Eingänge der Steuerung, die Fehlerinjektion und die Antriebsalarme.

Die Leistungskette

Die Kette der MC-1, vom Netz bis zur Spindel:

- **Hauptschalter QF1** und das **Netzteil 24 V** für die Steuerung;
- die **Steuerung**: ihr Zustand oder ihr Alarm;
- **Not-Halt SB0** und **KM1**, das Schütz, das er öffnet;
- das **Netzteil 48 V** für die Antriebe;
- die Antriebe **X**, **Y** und **Z** und ihre Achsen;
- der **Frequenzumrichter**, die **Spindel** und die **Kühlmittelpumpe**.

Die Eingänge der Steuerung

Eine Lampe für jeden Eingang: die Endschalter (**Endschalter X−**, **Endschalter X+**, **Endschalter Y−**, **Endschalter Y+**, **Endschalter Z**), der **Werkzeuglängensensor**, der **Not-Halt** und der **Antriebsalarm**. Eine Lampe ist grau, wenn der Eingang aus ist, und gelb oder rot, wenn er an ist.

Die Lampe **Werkzeuglängensensor** ist der Messtastereingang der Steuerung: Sie leuchtet auch, wenn der Messtaster etwas berührt.

Die Antriebe

Jeder Antrieb zeigt **Bereit** und seinen Schleppfehler oder **Gesperrt**, **Alarm**, **Startet** oder **Aus**. Die Details nennen die Spannungsversorgung der Antriebe und den Schleppfehler jeder Achse mit seinem Höchstwert.

Der Spindelantrieb (Frequenzumrichter)

Der **Frequenzumrichter** zeigt seine Ausgangsfrequenz und seinen Strom oder seinen Fehler: **OC** (Überstrom), **OL** (Überlast) oder **LU** (Unterspannung). Die Details nennen die Spindeldrehzahl und die Solldrehzahl, die Last und die Pumpe. Eine Spindel, die ohne ihr Kühlmittel lief, wird unter den Warnungen aufgeführt, mit der Dauer des Trockenlaufs.

Fehler in den Zwilling injizieren

Mit der Fehlerinjektion üben Sie, Fehler am Zwilling zu finden und zu beheben; verändert wird die Maschine in Ihrem Fenster, keine echte.

- **Leitungen**: Setzen Sie jede Leitung der Maschine (die Endschalter, den Werkzeuglängensensor, die Alarmkette der Antriebe, die Not-Halt-Leitung, den KM1-Stromkreis, den RS485-Bus, das Fehlerrelais des Frequenzumrichters) auf **Unterbrechung**, **Kurzschluss**, **Wackelkontakt** oder **Störimpulse** (ein einzelner Aussetzer von 50 µs). **Normal** setzt die Leitung wieder in Ordnung.
- **Antriebsalarm**: **X**, **Y** oder **Z** lässt den betreffenden Antrieb einen Fehler melden. Der Alarm eines Antriebs erlischt nur, wenn der Antrieb stromlos wird: Folgen Sie den Schritten zurück in den Betrieb in der Maschinenleiste.

Jeder injizierte Fehler wird unter den Warnungen aufgeführt, in die Konsole geschrieben und im Prüfprotokoll aufgezeichnet. Nach einem Antriebsalarm nennt die Ursache in der Maschinenleiste den von Ihnen injizierten Fehler und die Uhrzeit.

Alarme: jeder einzelne und seine Behebung

Jeder Alarm, was er bedeutet und seine Ursachen: Alarme. Die Maschinenleiste nennt die Schritte zurück in den Betrieb für den Alarm, den Sie haben (siehe Die Schritte zurück in den Betrieb):

Alarm	Schritte zurück in den Betrieb
Der Fehler eines Antriebs (ALARM:17 mit einem Antrieb im Alarm)	Not-Halt drücken, entriegeln, **Einschalten**, **Reset**, **Referenzfahrt**
Not-Halt gedrückt (ALARM:10)	Not-Halt entriegeln, **Einschalten**, **Reset**, **Referenzfahrt**
ALARM:17 bei wiederhergestellten Antrieben	**Reset**, **Referenzfahrt**
Referenzfahrt nötig (ALARM:11)	**Referenzfahrt**
Hard-Limit (ALARM:1)	**Reset**, **Entsperren**, im Tippbetrieb vom Endschalter wegfahren, **Referenzfahrt**
ALARM:3, 8, 9	**Reset**, **Referenzfahrt**
Jeder andere Alarm	**Reset**, **Entsperren**

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Ein Programm im Zwilling ausführen

Entwurf (Runde 14): Der Text wird noch geschrieben; die Abschnitte können noch wachsen.

Was der Zwilling ist

Die Maschine im Fenster ist ein digitaler Zwilling der MC-1: In ihm läuft Twinwrights eigener Kernel, derselbe Code wie auf der Maschine, und ein physikalisches Modell steht für die Motoren, die Antriebe, die Spindel, die Schalter und den Schnitt. Ein Programm läuft im Zwilling so, wie es auf der Maschine laufen würde, und der Zwilling zerspant sein eigenes Werkstück. Jede Zahl, die er über die physische Welt angibt, ist eine Schätzung.

Bevor Sie ausführen

Der Zwilling prüft ein Programm gegen die Aufspannung, die Sie ihm vorgeben; richten Sie ihn also so ein wie die echte Maschine:

- die Maschine (**Datei > Maschine wechseln…**);
- das Rohteil und wie es gespannt ist (der Arbeitsbereich **Vorbereiten**);
- die Werkzeuge (die **Werkzeugtabelle** im Dock);
- der Werkstücknullpunkt (**Maschine > Werkstücknullpunkt**).

Datei > Beispielteile… legt all das fest und lädt in einem Zug ein Programm.

> **Achtung**
>
> Eine Prüfung ist nur so gut wie die Aufspannung: Ein anderes Rohteil, Spannmittel, Werkzeug oder ein anderer Werkstücknullpunkt an der echten Maschine ist nicht das, was der Zwilling geprüft hat.

Ein Programm ausführen

1. Öffnen Sie das Programm: **Datei > Programm öffnen…**
2. Prüfen Sie es zuerst: **Programm > Programm prüfen** schickt es durch den Prüfmodus der Steuerung, sodass die Steuerung jeden Satz liest und die Maschine sich nicht bewegt.
3. Führen Sie es aus: **Zyklusstart** (F5). **Vorschub halt** (F6) pausiert es, **Zyklusstart** setzt fort, **Stopp** (F7) stoppt es. **Einzelsatz** führt es Satz für Satz aus, **Nächster Satz** führt den nächsten Satz aus.

Der Zwilling läuft in Echtzeit, sofern Sie nichts anderes wählen: **Simulation > Geschwindigkeit** lässt ihn mit halber bis 200-facher Echtzeit laufen oder mit **Maximal** – so schnell, wie dieser Computer kann –, und **Simulation > Simulation pausieren** hält die Zeit des Zwillings an (der Zustand der Maschine ändert sich nicht, solange sie steht). Ein Klick auf die Uhr rechts in der Maschinenleiste öffnet dieselbe Liste.

Ab ×100, bei **Maximal** und während der Berechnung bis zum Ende läuft der Zwilling **Reduziert** (die Kennzeichnung neben der Uhr): Die Ratterprüfung und die Kräfte können etwas später in den Kurven erscheinen, das Bild verwendet gröbere Zellen und eine dünnere Spur. Was das Programm tut – das abgetragene Material, Kollisionen, Software-Endschalter, Alarme –, wird wie bei ×1 beurteilt, und Ergebnis und Bericht sind dieselben. **Einstellungen > Allgemein > Volle Genauigkeit auch bei hoher Geschwindigkeit (langsamer)** schaltet **Reduziert** ab.

Wenn der Computer nicht mithalten kann, zeigt die Uhr in Gelb an, wie schnell der Zwilling wirklich läuft; ihr Tooltip nennt den Teil des Zwillings, der gerade am meisten Zeit braucht (zum Beispiel den Materialabtrag auf feinen Zellen), und was zu tun ist.

Simulation > Geschwindigkeit > Bis zum Ende berechnen (ohne Wiedergabe) führt das geladene Programm bis zu seinem Ende aus, ohne es Bild für Bild zu zeigen: Die Maschinenleiste zeigt, wie weit es ist, und die verbleibende Zeit; die 3D-Ansicht bleibt auf dem Anfangsstand und ändert sich, wenn es fertig ist, und der letzte Lauf im Arbeitsbereich **Analyse** wird wie gewohnt ausgefüllt. **Stopp** neben dem Programm hält es dort an, wo es gerade ist. Mit einer echten Maschine verbunden (Schattenmodus) ist nur ×1 möglich: Der Zwilling folgt der Zeit der Maschine.

Was der Zwilling prüft

Was sich aus dem Programm und der Geometrie der Aufspannung ergibt:

- die eigenen Alarme und Fehler der Steuerung, so wie die Steuerung der Maschine sie ausgibt: Software-Endlagen, Endschalter, eine Antastbewegung, die nichts berührt;
- Kollisionen von Werkzeughalter, Spannmutter und Spindel mit dem Rohteil, dem Spannmittel und dem Tisch sowie Beinahekollisionen;
- Schneiden im Eilgang (`G0`) und Schneiden bei stehender Spindel;
- Schneiden durch den Boden des Rohteils in das Spannmittel oder den Tisch;
- ein Werkzeug, das klemmen oder brechen würde: ein für das Werkzeug zu hoher Vorschub pro Zahn, eine Biegespannung am Grund der Spannuten, die über dem liegt, was das Werkzeug aushält;
- der Z-Freiraum: ob jedes Werkzeug und der Messtaster die Aufspannung bei ganz oben stehender Z-Achse frei passiert (siehe MC-1-Hardware);
- mit der Konstruktion des Bauteils in der Aufspannung: am Bauteil verbliebenes Material und zu tiefe Schnitte (**Ansicht > Einfärbung > Werkstück: Abweichung vom Bauteil**).

Die Befunde erscheinen im Reiter **Meldungen** des Docks, sobald sie auftreten.

Was der Zwilling schätzt

Was das physikalische Modell errechnet:

- die Schnittkräfte, das Drehmoment und die Leistung sowie die Last der Spindel: Die Spindel wird unter Last langsamer, wie eine echte, und zu viel Last löst ihren Antrieb aus;
- wie das Werkzeug und die Maschine unter dem Schnitt nachgeben, in der geschnittenen Form;
- Rattern: der Ratterkennwert entlang der Bahn (**Ansicht > Einfärbung > Spur: Ratterkennwert**) und die Spindeldrehzahlen, bei denen der Schnitt stabil ist (der Arbeitsbereich **Analyse**: **Stabilitätskarte berechnen**);
- der Rundlauffehler des Fräsers: Nuten und Bohrungen werden um diesen Betrag breiter;
- die Maschinenzeit, aus der eigenen Planung der Steuerung und der Bewegung der Antriebe;
- die Restkämme, die ein Kugelfräser zwischen seinen Bahnen hinterlässt (ihre Form, nicht die Rauheit der Oberfläche).

Die Spur lässt sich auch nach Vorschubgeschwindigkeit, Schnittkraft oder Spanungsdicke einfärben (**Ansicht > Einfärbung**), und der Reiter **Kurven** des Docks stellt den Lauf über der Zeit dar.

> **Hinweis**
>
> Die Schätzungen verwenden typische Parameter – veröffentlichte Schnittdaten, scharfe Hartmetallwerkzeuge, die Konstruktionswerte der Maschine –, bis Ihre Maschine vermessen ist. Verwenden Sie sie, um Programme zu vergleichen und Probleme vor dem Fräsen zu finden, nicht als Messung des Bauteils.

Was der Zwilling nicht modelliert

- die Textur und Rauheit der Oberfläche;
- Aufbauschneide, Späne und ihre Abfuhr;
- dünne Wände, die sich unter dem Schnitt durchbiegen;
- die Eigenwärme des Werkstücks;
- Werkzeugverschleiß: Geschnitten wird immer mit einem scharfen Werkzeug. Der Zwilling führt für jedes Werkzeug eine Schätzung seines Verschleißes – wie lange seine Schneiden im Material waren, welchen Weg sie geschnitten haben, welcher Anteil einer geschätzten Standzeit verbraucht ist (Taylor-Gleichung; die Werte der meisten Werkstoffe sind Schätzungen) – und meldet sie (tool_wear von machine_status über MCP), ändert aber den Schnitt nicht; ein neues Werkzeug derselben Spezifikation beginnt von vorn (setup_tool mit new). Werkzeuge, die Aluminium zerspanen, werden meist zuerst gewechselt, weil Aluminium an der Schneide haftet und das Bauteil Grate oder eine schlechtere Oberflächengüte zeigt;
- was nach einem Werkzeugbruch geschieht;
- Details, die feiner sind als das Raster der Programmprüfung: Bevor etwas läuft, betrachtet die Programmprüfung das Werkstück in Zellen von etwa 0,1 mm, feiner, wenn das Programm ein Werkzeug unter 2 mm Durchmesser hat: ein Zwanzigstel seines Durchmessers, bis hinunter zu 0,05 mm (gröber bei Rohteilen über 100 × 100 mm: höchstens eine Million Zellen, vier Millionen mit einem solchen Werkzeug), sodass ein Eilgang durch eine dünnere Wand oder ein Werkzeug, das eine Wand um weniger streift, die Prüfung unbemerkt passieren kann. Lassen Sie das Programm im Zwilling laufen, um es zu sehen: Der Zwilling schneidet selbst in Zellen von 0,02 mm.

Kunststoffe und Mikrowerkzeuge

Beim Fräsen von PMMA (Acryl) mit Werkzeugen von 0,1–0,5 mm – zum Beispiel für mikrofluidische Kanäle – lässt der Zwilling außerdem weg:

- die Temperatur des Schnitts: PMMA erweicht gegen seinen Glasübergang (etwa 105–115 °C), und ab etwa 70 °C verschweißen und verschmieren seine Späne am Werkzeug und im Kanal. Halten Sie Zahnvorschub und Drehzahl in dem Bereich, den der Werkzeughersteller angibt;
- den Luftstrahl oder das Kühlmittel und ihre Wirkung auf die Wärme und die Späne;
- Grate entlang der Kanten von Kanälen und Taschen;
- dünne Wände: In PMMA biegen sich Wände unter etwa 0,15 mm Dicke unter dem Schnitt oder brechen; der Zwilling schneidet die Form, die die Bahn des Werkzeugs und seine Nachgiebigkeit hinterlassen, nicht die Durchbiegung der Wand selbst;
- die Dämpfung des Schnitts selbst: Ein so kleines Werkzeug biegt sich bei einigen zehn Kilohertz und schneidet langsam, und seine Freifläche, die an der geschnittenen Fläche reibt, dämpft seine Schwingung. Der Zwilling lässt das weg, sodass sein Ratterkennwert für solche Werkzeuge auf der sicheren Seite liegt.

Die Modelle und was sie annehmen: Zwillingsmodelle.

Die Ergebnisse und der Bericht

- Das Werkstück in der 3D-Ansicht ist das, was der Zwilling geschnitten hat; seine Einfärbung zeigt die Tiefe des Schnitts oder, mit der Konstruktion des Bauteils, die Abweichung davon.
- Der Reiter **Meldungen** des Docks listet die Befunde der Prüfungen auf; sein Reiter **Kurven** die Kräfte, die Last und die Geschwindigkeiten über der Zeit. **Datei > Kurven exportieren (CSV)…** speichert sie.
- Ein KI-Client liest dieselben Ergebnisse über die MCP-Werkzeuge von Studio: siehe die Referenz MCP-Werkzeuge.

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Prüfung, Berichte und Einrichteblätter

Nachdem ein Teil gefräst ist, misst der Messtaster es im Vergleich zur Konstruktion. Studio schreibt, was gemessen wurde, zusammen mit der Schätzung des Zwillings in einen Prüfbericht. Bevor ein Teil gefräst wird, sagt ein Einrichteblatt dem Bediener, wie der Auftrag einzurichten ist. Beides ist in der Pro-Edition enthalten.

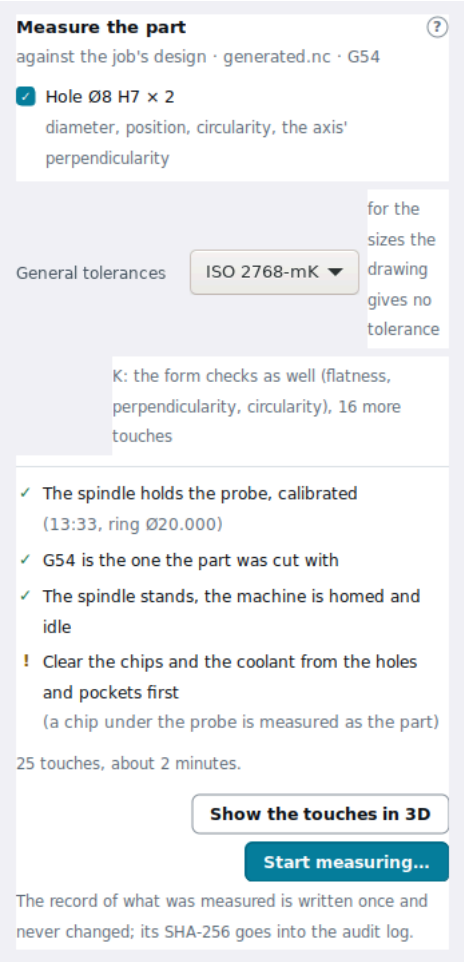
Ein Teil mit dem Messtaster messen

Messen Sie ein Teil dort, wo es gefräst wurde, bevor Sie es von der Maschine nehmen. Die Karte **Teil messen** befindet sich in **Vorbereiten**, im Schritt **Nullpunkt**, unter der Messtasterkarte. Sie vergleicht das Teil mit der Konstruktion seines CAM-Auftrags: Größe, Passung und Toleranz jedes Formelements. In 1.0 misst die Karte am Zwilling, so wie die Messtasterkarte. Folgt der Zwilling einer realen Maschine, messen Sie dort mit der eigenen Steuerung der Maschine.

Sie brauchen:

- Das gefräste Teil, noch so gespannt, wie es gefräst wurde. Verschieben Sie den Werkstücknullpunkt nicht: Die Positionen werden von ihm aus gemessen.
- Den Messtaster in der Spindel, an seiner Ringlehre kalibriert. Siehe Antasten.
- Den CAM-Auftrag des Programms: ein Programm, das in **CAM** oder vom KI-Assistenten erstellt wurde.

> **Hinweis**
>
> Das Messen ist in der Pro-Edition enthalten. In den anderen Editionen trägt die Karte die Kennzeichnung **Pro**, und **Messen starten…** sagt, wie Sie es ausprobieren können. Zuvor erstellte Datensätze und Berichte lassen sich in jeder Edition öffnen.



So messen Sie:

1. Klicken Sie in **Vorbereiten** auf den Schritt **Nullpunkt**. Ergebnis: Die Karte sagt, womit sie das Teil vergleicht: das Programm und die Nullpunktverschiebung, mit der es gefräst wurde, zum Beispiel G54.
2. Wählen Sie die Formelemente, die gemessen werden sollen. Zunächst sind alle gewählt. Ergebnis: Jedes Formelement sagt, was der Messtaster an ihm misst. Ein Formelement, das der Messtaster nicht messen kann, ist grau und sagt, warum.
3. Wählen Sie unter **Allgemeintoleranzen** die Klasse für die Maße, die keine eigene Toleranz haben. Hängen Sie H, K oder L für die Form- und Lageprüfungen an (siehe Toleranzen). Ergebnis: Die Karte sagt, wie viele Antastungen es sind und wie lange sie etwa dauern.
4. Lesen Sie die Prüfungen: Der Messtaster steckt kalibriert in der Spindel, der Werkstücknullpunkt ist der, mit dem das Teil gefräst wurde, die Spindel steht, die Maschine ist referenziert und bereit.
5. Entfernen Sie Späne und Kühlmittel aus Bohrungen und Taschen. Ein Span unter dem Messtaster wird als Teil gemessen.

6. Um zu sehen, wo der Messtaster antasten wird, klicken Sie auf **Antastpunkte in 3D zeigen**. Ergebnis: Die 3D-Ansicht zeigt die Antastpunkte von der Seite und die Antastpunkte nach unten.
7. Klicken Sie auf **Messen starten…**. Ergebnis: Der Messtaster tastet jedes Formelement an, und die Karte sagt, was gerade gemessen wird.

> ⚠ **Warnung**
>
> Der Messtaster fährt in der Nähe des Teils mit Vorschubgeschwindigkeiten. Halten Sie Hände und Werkzeuge von der Maschine fern, solange sie misst.

Was gemessen wird:

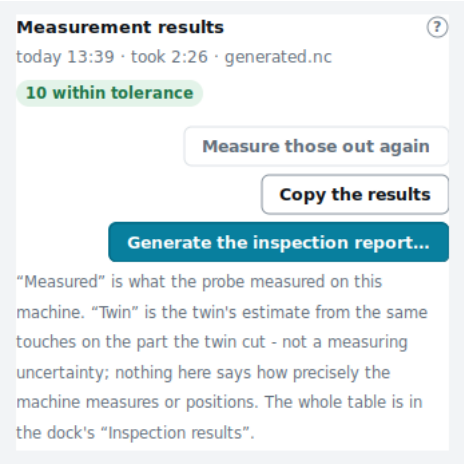
Formelement	Prüfungen
Tasche, Kontur	Größe und Lage der Wände, Tiefe des Bodens
Bohrung, Zapfen	Durchmesser und Lage
Nut	Breite und Tiefe
Planfläche	Tiefe
Fase	Breite, Tiefe und Winkel
V-Nut (Gravur oder V-Gravur mit einem V-Fräser)	Breite, Tiefe, Winkel und Versatz von ihrer Linie
Gravur mit einem Schaftfräser, Gewinde, 3D-Fläche	Nicht gemessen

Die Form- und Lageprüfungen ergänzen Ebenheit, Rechtwinkligkeit, Rundheit und Geradheit, wo sie anwendbar sind (siehe Form).

Wenn der Messtaster fertig ist, tritt die Karte **Messergebnisse** an die Stelle der Messkarte. Sie zeigt:

- wann das Teil gemessen wurde, wie lange die Messung gedauert hat und das Programm;
- wie viele Prüfungen in Toleranz sind, wie viele außer Toleranz und wie viele nicht beurteilt werden: **im Zwilling nicht beurteilt** (seine Zellen sind dort gröber als ein Viertel der Toleranz), **ohne Toleranz** oder **nicht gemessen**;
- jede Prüfung außer Toleranz mit ihrem Messwert, ihrem Grenzwert und der Angabe, um wie viel sie darüber oder darunter liegt;
- für das erste Formelement außer Toleranz die wahrscheinlichste Ursache und was als Nächstes zu tun ist.

In 1.0 misst der Messtaster das Teil, das der Zwilling gefräst hat; die Ursache kommt deshalb allein aus dem Zwilling. Ist das Teil des Zwillings an denselben Antastpunkten selbst außer Toleranz, fräst das Programm es so: Ändern Sie es in **CAM**. Andernfalls unterscheiden sich Messwert und Schätzung des Zwillings nur durch das, was der Zwilling vom Messtaster modelliert (den wirksamen Kugelradius und, bei einem typischen Messtaster, den Vorlaufweg, den eine Kalibrierung an der Ringlehre herausnimmt), und durch die Größe der Zellen des Zwillings dort, wo der Messtaster antastet. Kalibrieren Sie den Messtaster an der Ringlehre und messen Sie erneut. Die Ursachen sind Vorschläge, keine Diagnose.



Die Tasten der Karte:

- **Außer Toleranz erneut messen**: misst die Prüfungen außer Toleranz noch einmal, zum Beispiel nachdem Sie den Messtaster neu kalibriert haben.
- **Ergebnisse kopieren**: kopiert die Tabelle als Text, die Spalten durch Tabulatoren getrennt, für eine Tabellenkalkulation.
- **Prüfbericht erstellen…**: schreibt den Bericht (siehe Der Prüfbericht).

Die ganze Tabelle steht im Reiter **Messergebnisse** des Docks. Jede Zeile ist eine Prüfung, mit den Spalten **Soll**, **Toleranz**, **Gemessen**, **Abweichung** und **Zwilling** (die Schätzung des Zwillings) und der Bewertung: ✓ oder *X* mit der Angabe, wie weit sie außer Toleranz ist. Zeilen außer Toleranz sind rot, und die 3D-Ansicht markiert die Ergebnisse am Teil.

Check	Desi…	Tolerance	Measu…	Deviation	Twin	
Hole 1 diameter	Ø8	H7 +0.015/0	8.001	+0.001	8.004	✓
Hole 1 position X	8.000	±0.2 (m)	8.000	+0.000	8.001	✓
Y	32.000	±0.3 (m)	32.000	+0.000	32.000	
Hole 1 circularity	○	0.015 (K)	0.002		0.003	✓
Hole 1 axis' perpendicularity	⊥	0.4 (K)	0.000		0.001	✓
Hole 2 diameter	Ø8	H7 +0.015/0	8.001	+0.001	8.004	✓
Hole 2 position X	52.000	±0.3 (m)	52.001	+0.001	52.001	✓
Y	32.000	±0.3 (m)	32.001	+0.001	32.000	
Hole 2 circularity	○	0.015 (K)	0.002		0.003	✓
Hole 2 axis' perpendicularity	⊥	0.4 (K)	0.000		0.001	✓

Jede Prüfung wird als Datensatz festgehalten. Der Datensatz wird einmal geschrieben und nie geändert, und sein SHA-256-Hash kommt ins Prüfprotokoll (siehe Eine Kopie mit dem Prüfprotokoll abgleichen).

Der KI-Assistent misst auf dieselbe Weise, mit dem Tool `inspect_part`. Bitten Sie ihn, zum Beispiel: „Prüfe das Teil von bracket.nc mit dem Messtaster".

1. Der Assistent sagt, was er messen wird und wie viele Antastungen und Sekunden die Form- und Lageprüfungen zusätzlich brauchen.
2. Bestätigen Sie, bevor der Messtaster fährt. Ergebnis: Die Antwort listet jede Prüfung mit ihrem Messwert, der Schätzung des Zwillings und der Bewertung auf.

Toleranzen

Jede Prüfung wird an einer Toleranz bewertet. Die eigene Toleranz oder Passung eines Maßes stammt aus dem CAM-Auftrag, so wie sie im Feld des Maßes eingegeben wurde (zum Beispiel 8 H7). Die übrigen Maße erhalten die Allgemeintoleranz.

Allgemeintoleranzen (ISO 2768-1)

ISO 2768-1 kennt vier Klassen: f (fein), m (mittel), c (grob) und v (sehr grob). Ein Maß ohne eigene Toleranz erhält die Toleranz der Klasse für seinen Nennmaßbereich. Wählen Sie die Klasse auf der Karte, unter **Allgemeintoleranzen**. Oder bitten Sie den Assistenten darum, zum Beispiel: „Prüfe es nach ISO 2768-m".

Form: Ebenheit, Rechtwinkligkeit, Rundheit, Geradheit (ISO 2768-2)

Hängen Sie H, K oder L an die Klasse an, zum Beispiel „mK". Der Messtaster prüft dann auch die Form:

Prüfung	So wird gemessen
Ebenheit	Ein Boden, in einem 3 × 3-Raster angetastet
Rechtwinkligkeit	Eine Wand oder die Achse einer Bohrung, in einer zweiten Tiefe erneut angetastet
Rundheit	Eine Bohrung oder ein Zapfen, an 8 Punkten angetastet
Geradheit	Eine lange Wand, an 3 bis 8 Punkten angetastet

Die Karte oder der Assistent sagt vor der Fahrt des Messtasters, wie viele Antastungen die Form- und Lageprüfungen zusätzlich brauchen.

> **Hinweis**
>
> Ein Messtaster, der nicht in jeder Richtung kalibriert ist, hat sein eigenes Lobing. Der Bericht gibt neben jeder Rundheit an, wie groß das Messtastermodell des Zwillings es annimmt.

Passungen (ISO 286)

Ein Maß, das mit seiner Passung geschrieben ist, zum Beispiel Ø8 H7, wird am Toleranzfeld der Passung bewertet. Siehe Passungen und Toleranzen.

Fasen und V-Nuten

Breiten, Tiefen und Winkel von Fasen sowie Breiten, Tiefen und Winkel von V-Nuten erhalten die Klasse f nach ISO 2768-1, sofern Sie keine eigene Toleranz angeben.

Was der Zwilling schätzt

Der Zwilling führt dieselben Antastungen an seinem simulierten Werkstück aus. Seine Schätzung ist das, was das Programm im Zwilling gefräst hat. Weichen der Messwert und die Schätzung voneinander ab, sehen Sie sich die Maschine und die Aufspannung an: die Länge des Werkzeugs, den Werkstücknullpunkt, das Spannmittel, ein verschlissenes Werkzeug.

Die Schätzung des Zwillings ist eine Berechnung aus seinen physikalischen Modellen. Sie ist nicht die Messunsicherheit der Maschine.

Der Prüfbericht

Jede Prüfung wird als Datensatz festgehalten. Aus einem Datensatz schreibt Studio den Bericht:

- eine CSV-Datei (UTF-8) für eine Tabellenkalkulation;

- eine PDF-Datei zum Drucken und Ablegen.

Der Bericht listet den Auftrag, die Aufspannung, die Werkzeuge, den Messtaster und seine Kalibrierung auf und für jedes Formelement: das Nennmaß, die Toleranz und ihre Grundlage, die Schätzung des Zwillings, den Messwert, die Abweichung und die Bewertung. Er nennt den Grund, warum ein Formelement nicht gemessen wurde, und zeigt das Werkstück des Zwillings von oben.

Bitten Sie den Assistenten, ihn zu schreiben, zum Beispiel: „Schreibe den Bericht der letzten Prüfung". Derselbe Datensatz ergibt dieselben Dateien, Byte für Byte.

Wo die Berichte liegen

Prüfberichte und Einrichteblätter liegen im Berichtsordner:

- Windows: Dokumente\Twinwright Studio\Reports
- Linux: ~/Documents/Twinwright Studio/Reports

Wenn Sie den Programmordner in den Einstellungen verschoben haben, liegt der Berichtsordner darin.

Eine Kopie mit dem Prüfprotokoll abgleichen

Studio hält den SHA-256-Hash jedes Datensatzes, jedes Berichts und jedes Einrichteblatts in seinem Prüfprotokoll fest. Jeder Eintrag ist mit dem vorherigen verkettet. Eine Kopie, die jemand Ihnen schickt, können Sie so prüfen:

- Fragen Sie den Assistenten: „Prüfe diese Datei gegen das Prüfprotokoll", mit dem Pfad der Datei.
- Oder führen Sie auf dem Computer, auf dem sie geschrieben wurde, aus:

```
mc1sim --inspect-verify <file>
```

Die Antwort sagt, ob das Prüfprotokoll den SHA-256-Hash der Datei enthält, wann sie geschrieben wurde und ob die Kette des Prüfprotokolls unversehrt ist. Eine Kopie, die um ein einziges Bit verändert wurde, wird nicht gefunden.

Einrichteblätter

Ein Einrichteblatt ist eine Seite für den Bediener an der Maschine. Es sagt, was einzurichten ist und in welcher Reihenfolge. Studio schreibt eines für jeden Auftrag, in der Sprache, die Sie verwenden. Das Papier ist A4, in den USA und Kanada Letter.

Bitten Sie zum Schreiben den KI-Assistenten, zum Beispiel: „Schreibe das Einrichteblatt für bracket.nc". Der Assistent verwendet das Tool `setup_sheet`.

> **Hinweis**
>
> Einrichteblätter sind in der Pro-Edition enthalten.

Was ein Einrichteblatt zeigt

- Programme: Name, Version und Anzahl der Sätze jedes Programms sowie die ersten 8 Hexadezimalziffern seines SHA-256-Hashs. Vergleichen Sie sie mit dem Programm, das in der Maschine geladen ist.
- Rohteil und Spannmittel: Größe und Material des Rohteils und wie der Zwilling es aufspannt. Bei einem Schraubstock: die Parallelunterlagen, die Spanntiefe und wie weit das Rohteil über die Backen übersteht.
- Werkzeuge: Nummer, Art, Durchmesser, Schneiden, Auskraglänge und Spezifikation jedes Werkzeugs sowie die Bearbeitungen, die es verwenden.
- Jede Aufspannung: Wie das Teil dorthin kommt (umgedreht, anders gespannt), wo der Werkstücknullpunkt liegt und welche Antastzyklen ihn finden. Die Zyklen tragen die Namen, die sie im Messtaster-Panel haben.
- Verfahrbereich: Der niedrigste und der höchste X-, Y- und Z-Wert, die das Programm anfährt, in Werkstückkoordinaten.
- Z-Freiraum: Siehe Z-Freiraum.
- Bearbeitungen: Strategie jeder Bearbeitung und wie sie ins Material eintaucht, Werkzeug, Drehzahl, Vorschub und Tiefe sowie die benötigte Zeit.
- Draufsicht: Das Rohteil, die Schraubstockbacken, der Werkstücknullpunkt und der Umriss jedes Formelements.
- Anmerkungen: Die Anmerkungen des CAM, auf Englisch.

Die Zeiten werden aus dem Programm geschätzt: aus seinen Vorschüben sowie der Eilganggeschwindigkeit und der Beschleunigung der Maschine. Werkzeugwechsel werden nicht gezählt. Ein Probelauf im Zwilling liefert Zeiten, in denen Beschleunigung und Verzögerung berücksichtigt sind.

Z-Freiraum

Für jede Aufspannung prüft das Blatt, ob der Messtaster und jedes Werkzeug über die Aufspannung hinwegkommen. Es prüft außerdem, ob jedes Werkzeug den höchsten und den tiefsten Punkt des Programms erreicht. Die Höhen sind die des Z-Budgets des Zwillings, von der Tischoberfläche aus gemessen.

Prüfpunkt	Muss erreichen
Messtaster	1 mm über dem höchsten Punkt der Aufspannung, bei Z nahe dem oberen Ende seines Verfahrwegs
Jedes Werkzeug, oben	0,5 mm über dem höchsten Punkt der Aufspannung und den höchsten Punkt des Programms
Jedes Werkzeug, unten	Den tiefsten Punkt seiner eigenen Bearbeitungen

Jede Zeile zeigt **Ausreichend** oder, in Rot, wie viele Millimeter fehlen.

> **Achtung**
>
> Eine rote Zeile bedeutet, dass das Programm mit dieser Aufspannung nicht laufen kann. Verwenden Sie ein Werkzeug mit weniger Auskraglänge oder einen kürzeren Messtaster, oder spannen Sie das Teil tiefer, zum Beispiel mit Spannpratzen.

Eine Spannpratze kann höher stehen als der höchste Punkt des Programms. Das Blatt sagt das dann: Das Programm darf die Spannpratze nicht kreuzen. Ein Probelauf im Zwilling prüft das.

Drucken und Unterschreiben

1. Öffnen Sie das Einrichteblatt in einem Webbrowser.
2. Drucken Sie es aus. Ergebnis: Jede Aufspannung nach der ersten beginnt auf einer neuen Seite.
3. Füllen Sie an der Maschine das Feld am Ende aus: den Bediener, das Datum und die Erstmusterprüfung (bestanden oder nicht bestanden und die Nummer ihres Prüfberichts).

Ein neues Einrichteblatt desselben Auftrags ersetzt die Datei. Sein SHA-256-Hash wird im Prüfprotokoll festgehalten.

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Eine echte Maschine verbinden

Entwurf (Runde 14): Der Text wird noch geschrieben; die Abschnitte können noch wachsen.

Zwei Arten, mit einer Maschine zu arbeiten

- Steuern Sie die MC-1 mit einem eigenen Sender-Programm (bCNC, gSender oder Universal G-code Sender) an ihrem USB-Anschluss. Siehe Sender-Software.
- Stellen Sie den Zwilling im Schattenmodus (Vorschau) daneben: Ihr Sender-Programm verbindet sich statt mit der Maschine mit Studio, Studio reicht jeden Satz an die Maschine weiter, lässt dieselben Sätze im Zwilling laufen und vergleicht beide.

Ein Programm von Studio selbst an eine echte Maschine zu senden, ist in 1.0 nicht enthalten.

Schattenmodus (Vorschau)

> **Hinweis**
>
> In 1.0 ist der Schattenmodus eine Vorschau, die Sie über die Befehlszeile starten. Er wurde mit einem zweiten Zwilling getestet, der die Maschine vertritt, noch nicht mit einer echten. Im Fenster gibt es noch keine Stelle, an der Sie den Anschluss wählen, die Verbindung herstellen oder den Vergleich ansehen können; das kommt in einer späteren Version.

Schattenmodus starten

Sie brauchen: die MC-1, eingeschaltet und per USB mit diesem Computer verbunden; Ihr Sender-Programm, installiert.

1. Suchen Sie den Anschluss der Maschine. Unter Windows: **Geräte-Manager > Anschlüsse (COM & LPT)**, zum Beispiel COM5.
2. Starten Sie Studio mit diesem Anschluss in einer Eingabeaufforderung:

   ```
   TwinwrightStudio.exe --shadow COM5
   ```

   Ergebnis: Studio öffnet den Anschluss, und der Zwilling folgt der Maschine.

3. Verbinden Sie sich in Ihrem Sender-Programm statt mit der Maschine mit dem Sender-Port von Studio, `localhost:2323`. Ergebnis: Die Sätze Ihres Sender-Programms gehen über Studio an die Maschine, und der Zwilling führt dieselben Sätze aus.

Der Sender-Port von Studio nimmt nur Verbindungen von diesem Computer an. Um ihn von einem anderen Computer aus zu erreichen, fügen Sie `--bind any` hinzu.

Was der Zwilling vergleicht

- Die Antworten, Satz für Satz: jedes `ok` oder jeder Fehler und die Alarme in ihrer Reihenfolge.
- Die Bahn: die Position, die die Maschine meldet, gegenüber der Bahn des Zwillings und wie weit der Zwilling voraus oder zurück liegt.
- Die Einstellungen: einmal, wenn die Maschine verbunden ist und bevor sich Ihr Sender-Programm verbindet. Verbinden Sie Ihr Sender-Programm, nachdem sich Studio mit der Maschine verbunden hat.
- Die Steuerung: einmal, zusammen mit den Einstellungen. Ob beide dieselbe Firmware laufen lassen und, mit Twinwrights eigenem Kernel auf beiden, seine Version und was die Platine der Maschine hat: Spindel, Encoder, Handrad, Achsen, Messtaster, Verbindung. Der Zwilling modelliert die Zusatzanschlüsse und die SD-Karte der Platine nicht: Der Vergleich listet sie auf, und sie zählen nicht als Unterschied.
- Die Laufzeiten der Teile des Programms.

Der Wächter

Starten Sie Studio zusätzlich mit `--shadow-guard`, und der Zwilling prüft jeden Satz Ihres Sender-Programms, bevor er die Maschine erreicht. Sagt der Zwilling einen Crash voraus, hält er den Satz zurück und setzt die Maschine auf Vorschub halt. Das Fenster zeigt den Zustand des Wächters unter **Diagnose**.

Was die Maschine nicht erreichen kann

Im Schattenmodus erreichen die Maschine nur die Sätze Ihres Sender-Programms, dazu die eigenen Abfragen des Zwillings, die nichts verändern (die Einstellungen und der Status).

- Die Bedienelemente des Fensters wirken nur auf den Zwilling, und die, die die Maschine bewegen oder starten würden, werden abgelehnt, solange der Zwilling einer Maschine folgt. Das Messtaster-Panel ist gesperrt.
- Ein KI-Client kann nur lesen: Er sieht den Zustand der Maschine und ändert nichts.
- Ein externes Handrad, das an Studio angeschlossen ist, wirkt nur auf den Zwilling.

> ⚠ **Warnung**
>
> Im Schattenmodus halten der Stopp und der Not-Halt des Fensters die Maschine nicht an. Um die Maschine anzuhalten, verwenden Sie ihren eigenen Not-Halt oder den Stopp Ihres Sender-Programms.

Sender-Software

Die mit Twinwrights eigenem Kernel getesteten Sender-Programme und wie Sie sie einrichten: Sender-Software.

Wenn etwas abweicht

Ein Unterschied bedeutet, dass das Profil oder die Einstellungen des Zwillings nicht die dieser Maschine sind oder dass die Maschine etwas tut, was das Modell nicht tut. Prüfen Sie zuerst die Einstellungen: Der Vergleich listet die auf, die sich unterscheiden.

Der KI-Assistent

Eine KI kann mit denselben Tools am Zwilling arbeiten wie Sie: Sie richtet ein, schreibt und prüft Programme und lässt sie in der Simulation laufen. Sie können sich mit einer KI im eigenen Assistenten von Studio unterhalten oder einen KI-Client wie Claude Desktop verbinden. Alles, was die Maschine im Fenster verändern würde, wartet auf Ihr Ja. Die KI ist nicht Teil einer Sicherheitsfunktion.

Die KI einschalten

In einer neuen Installation ist die KI aus. Um sie einzuschalten, wählen Sie einen dieser Wege:

- Klicken Sie in der oberen Leiste auf den Zustand der KI (**KI aus**) und wählen Sie **KI-Schalter des Fensters**;
- **Extras > Einstellungen… > KI und MCP**: **KI-Schalter**, dann **OK**;
- senden Sie eine Nachricht im Assistenten: Damit wird der Schalter eingeschaltet.

Studio behält den Schalter für den nächsten Start so bei, wie Sie ihn verlassen. Das Menü **KI** (**KI-Assistent…**, **KI-Aktivität…**, **KI-Clients verbinden…**) wird ab der Stufe Bediener angezeigt.

Bei ausgeschaltetem Schalter können KI-Clients weiterhin Übungsmaschinen im Hintergrund verwenden, aber nicht die Maschine in Ihrem Fenster.

Der KI-Dienst und sein Schlüssel

Der Assistent spricht mit einem KI-Dienst Ihrer Wahl, mit Ihrem eigenen Konto:

- **Anthropic (Claude)**, **OpenAI**, **DeepSeek**, **Qwen (Tongyi)**, **Kimi (Moonshot AI)**, **Zhipu (GLM)**;
- **LM Studio (lokal)** oder **Ollama (lokal)**: ein Modell auf Ihrem eigenen Computer, kein Schlüssel nötig;
- **Benutzerdefiniert (OpenAI-kompatibel)**: jeder Dienst mit einer OpenAI-kompatiblen Schnittstelle, unter der Adresse, die Sie eingeben.

So richten Sie ihn ein:

1. Öffnen Sie **KI > KI-Assistent…**
2. Wählen Sie den **Anbieter**. Studio trägt dessen **Basis-URL** ein.
3. Fügen Sie Ihren Schlüssel in **API-Schlüssel** ein und klicken Sie auf **Schlüssel speichern**. Ergebnis: Der Schlüssel wird verschlüsselt auf diesem Computer gespeichert. Unter Windows kann ihn nur Ihr Windows-Konto lesen. Unter Linux kommt er in den Schlüsselbund des Desktops (Secret Service: GNOME Keyring, KWallet); ohne ihn wird er mit einem Schlüssel in einer Datei verschlüsselt, die nur Ihr Benutzerkonto lesen kann, und die Zeile unter dem Schlüssel sagt das.
4. Klicken Sie auf **Modelle auflisten** und wählen Sie ein **Modell**.

Studio schreibt den Schlüssel weder in seine Einstellungen noch in die Registrierungsdatenbank noch in das Prüfprotokoll. Um einen gespeicherten Schlüssel zu löschen, leeren Sie das Feld und klicken Sie auf **Schlüssel speichern**.

> **Hinweis**
>
> Was Sie an den KI-Dienst senden, geht unter Ihrem Konto und nach dessen Bedingungen an diesen Dienst: Ihre Nachrichten und, bei eingeschaltetem **Programm, Aufspannung und Prüfergebnisse des Fensters anhängen**, das Programm, die Aufspannung und die Prüfergebnisse sowie die Ergebnisse der Tools, die die KI verwendet. Um alles auf Ihrem Computer zu behalten, verwenden Sie ein lokales Modell.

Vor Ihrer ersten Nachricht an einen Cloud-Dienst sagt der Assistent einmal, was dorthin geht. Klicken Sie auf **Senden**, um fortzufahren, oder auf **Lokales Modell verwenden…**, um zu LM Studio zu wechseln; Ihre Nachricht bleibt im Feld. **Extras > Einstellungen… > KI und MCP** listet die Dienste auf, denen Sie zugestimmt haben; **Erneut fragen** lässt den Assistenten erneut fragen.

Das Assistenten-Panel

KI > KI-Assistent… öffnet den Assistenten in einem eigenen Fenster.

- Tippen Sie in das Feld unten. Die Eingabetaste sendet; Umschalt+Eingabetaste beginnt eine neue Zeile. **Stopp** hält die Antwort an.
- **Programm, Aufspannung und Prüfergebnisse des Fensters anhängen (wenn geändert)** gibt der KI mit jeder Nachricht, was in Ihrem Fenster ist. Es ist standardmäßig eingeschaltet.
- Die Unterhaltung zeigt jedes Tool, das die KI verwendet, mit ✓, wenn es funktioniert hat, und ✗, wenn nicht.
- Bilder, die die KI erzeugt (das Werkstück, die Werkzeugbahn, die Abweichung vom Teil, ein Vergleich), erscheinen in der Unterhaltung. Doppelklicken Sie auf eines, um es zu vergrößern oder zu speichern.
- **Neue Unterhaltung** beginnt neu. Unterhaltungen werden nicht gespeichert, wenn Sie Studio schließen.

Das Schließen des Fensters blendet es nur aus; die Unterhaltung bleibt bestehen, bis Sie Studio beenden.

Was die KI kann und was nicht

Bei eingeschaltetem Schalter kann die KI:

- den Zustand der Maschinen lesen sowie referenzieren, entsperren, im Tippbetrieb verfahren und G-Code-Sätze senden;
- das Rohteil, die Werkzeuge, den Werkstücknullpunkt und das Teil einrichten, antasten und prüfen;
- Programme prüfen, simulieren, optimieren, generieren, bearbeiten und laden sowie einen Lauf starten oder neu starten;

- Läufe und ihre Ergebnisse lesen, vergleichen und Bilder davon erzeugen;
- das Prüfprotokoll und den Zustand der Lizenz lesen;
- Ihnen beim Erklären etwas im Fenster zeigen: einen Arbeitsbereich, die Blickrichtung der 3D-Ansicht, die Spur, den Programmweg, die Einfärbung, einen Reiter des Docks, einen Schritt von **Vorbereiten** – dabei ändert sich nur, was das Fenster zeigt;
- in diesem Handbuch suchen und daraus antworten;
- die Sprache, die Region, die Einheiten, die Währung und den Maschinenstundensatz ändern – erst nachdem Sie auf **Erlauben** geklickt haben (siehe unten), mit derselben Wirkung wie unter **Extras > Einstellungen...**.

Tools, die zu einer kostenpflichtigen Edition gehören (Prüfung, Berichte, Verschachtelung, der Optimierer), funktionieren nur in dieser Edition.

Die KI kann die übrigen Einstellungen von Studio nicht ändern: die Benutzerstufe, den KI-Schalter, die Art, wie Studio Sie fragt, den Datenschutz, das Protokoll oder die Darstellung; ebenso wenig die Lizenz oder das Prüfprotokoll.

Nachfragen, bevor sie die Maschine ändert

Bevor die KI die Maschine im Fenster ändert (im Menü der KI in der oberen Leiste und in **Extras > Einstellungen... > KI und MCP**) hat zwei Möglichkeiten:

- **Jedes Mal fragen** (die Standardeinstellung);
- **Nicht fragen (die KI bedient das Fenster direkt)**.

Wenn die KI die Maschine im Fenster ändern will (sie bewegen, einschalten, den Not-Halt entriegeln, ein Programm laden oder ausführen, die Aufspannung oder die Werkzeuge ändern) oder eine der Einstellungen, die sie ändern darf, öffnet sich eine Leiste über der Maschinenleiste. Sie sagt, welche KI fragt und was sie tun will, und zählt herunter.

1. Lesen Sie die Frage.
2. Klicken Sie auf **Erlauben** oder **Ablehnen**. Die Eingabetaste antwortet nicht: Verwenden Sie die Maus oder wechseln Sie mit Tab zu einer Taste und drücken Sie die Leertaste.

Keine Antwort innerhalb von 60 s zählt als **Ablehnen**. Ein Ja gilt zwei Minuten lang auch für die eigenen Folgeanfragen dieser Aktion.

Studio fragt nie nach, wenn die KI die Maschine stoppt, anhält oder zurücksetzt, den Not-Halt drückt oder nur liest. Bei Übungsmaschinen und den verborgenen Kopien, auf denen die KI simuliert, wird nicht nachgefragt.

Nur lesen an einer realen Maschine

Wenn der Zwilling im Fenster einer realen Maschine folgt (Schattenmodus, in 1.0 eine Vorschau; siehe Eine reale Maschine), liest und analysiert die KI nur. Sie kann die Maschine dort nicht bewegen, weder ihre Aufspannung noch ihre Nullpunktverschiebungen ändern noch Programme ausführen, und sie fragt nicht: Studio lehnt diese Anfragen ab. Übungsmaschinen arbeiten wie zuvor.

KI-Aktivität

KI > KI-Aktivität... zeigt, was KI-Clients auf diesem Computer tun:

- **Hintergrundmaschinen**: die Übungsmaschinen, die die KI verwendet und die Sie im Fenster nicht sehen („Fenster" ist die Maschine in Ihrem Fenster);
- **Läufe**: die letzten 16 Simulationen und Optimierungen;
- **Verbundene KI-Clients**: wer verbunden ist, wie, und ihr letzter Aufruf.

Von hier aus können Sie einen Lauf stoppen, das Bild eines Laufs ansehen, eine Hintergrundmaschine schließen und den Zwillings-Host stoppen oder starten. Der Zwillings-Host ist das Hintergrundprogramm, das die Übungsmaschinen enthält. Er startet, wenn sich ein KI-Client verbindet, und beendet sich nach 30 Minuten ohne Nutzung.

Der Reiter **KI-Aktivität** des Docks listet die Aufrufe der KI an der Maschine im Fenster auf: Zeit, Aufruf, Maschine, Wer und Ergebnis.

Andere KI-Clients verbinden (MCP)

KI-Clients, die MCP (Model Context Protocol) sprechen, können die Tools von Studio verwenden.

1. Öffnen Sie **KI > KI-Clients verbinden...**
2. Klicken Sie auf Ihren Client: **Claude Desktop**, **ChatGPT Desktop (Modi Codex und Work)**, **Cursor**, **VS Code (GitHub Copilot)** oder **LM Studio (lokale offene Modelle)**. Ergebnis: Für Claude Desktop und ChatGPT Desktop zeigt Studio die Änderung, die es an den Einstellungen des Clients vornehmen wird, und führt sie aus, wenn Sie auf **OK** klicken; die alte Datei des Clients bleibt daneben erhalten. Die anderen öffnen ihre eigene Installationsseite.
3. Beenden Sie den Client vollständig und öffnen Sie ihn erneut.
4. Schalten Sie in Studio den **KI-Schalter des Fensters** ein.

Für jeden anderen Client kopiert **Andere Clients: Konfiguration kopieren** die Einstellungen zum Einfügen in den Client. Der Client startet `TwinwrightStudio.exe --mcp` im Hintergrund; **Befehl und Konfiguration** zeigt den genauen Befehl.

Clients, die nur eine Adresse annehmen

Für Clients, die nur eine Adresse annehmen, schalten Sie im Menü der KI **Lokale URL (HTTP, für Clients, die nur eine URL annehmen)** ein und verwenden Sie **Lokale URL (HTTP) kopieren**. Die Adresse funktioniert nur auf diesem Computer und braucht das Token, das dazugehört. **Neues Token** macht das alte Token ungültig.

Die Tools, ihre Argumente und ihre Ergebnisse: MCP-Tools.

Was aufgezeichnet wird

Extras > Prüfprotokoll... (Strg+L) zeichnet auf, was die KI tut, mit dem Namen der KI in der Spalte **Wer**. Filtern Sie **Wer** nach **KI**, um nur die Einträge der KI zu sehen; doppelklicken Sie auf einen Aufruf, um alle seine Einträge zusammen zu sehen.

Auf der voreingestellten Stufe **Standard** hält das Prüfprotokoll fest:

- das Ein- und Ausschalten des KI-Schalters und das Ändern der Art, wie Studio fragt;
- jede Frage, Ihre Antwort und jede Aktion der KI an der Maschine im Fenster;
- jeden Tool-Aufruf mit einer Kurzform seiner Argumente;
- das Verbinden und Trennen von Clients;
- die Nachrichten des Assistenten: die ersten 180 Bytes in der Zusammenfassung, bis zu 2.000 Bytes im Eintrag.

Schlüssel, Token und Passwörter werden als `***` geschrieben.

Sprache und Einheiten

Studio folgt beim ersten Start der Sprache Ihres Systems. Sie können die Sprache, die Einheiten und die Darstellung von Zahlen, Datumsangaben und Geldbeträgen jederzeit ändern. Programme und Dateien bleiben, wie sie sind.

Die Sprache ändern

Wählen Sie **Extras > Language / 语言** und eine Sprache: English, 简体中文 (Chinesisch, vereinfacht), Français, Deutsch, 日本語 oder Español. Der Name des Menüs ist in jeder Sprache derselbe, sodass Sie es auch in einer Sprache finden, die Sie nicht lesen.

Das Fenster ändert sich sofort; Sie müssen Studio nicht neu starten. Zeilen, die bereits in der Konsole stehen, bleiben in der Sprache, in der sie geschrieben wurden.

Extras > Einstellungen… > Allgemein > Sprache bietet eine weitere Möglichkeit, **Wie Windows**: Studio folgt dann bei jedem Start der Sprache des Systems. Für eine Sprache, zu der Studio kein Paket hat, greift Studio auf Englisch zurück.

Einheiten: Millimeter und Zoll

Extras > Einheiten bietet **Wie in der Region üblich**, **Metrisch (mm)** oder **Zoll** zur Wahl. Die Änderung gilt sofort.

In Zoll zeigt Studio:

Größe	Metrisch	Zoll
Länge	mm	in
Vorschub	mm/min	in/min
Schnittgeschwindigkeit	m/min	ft/min
Kraft	N	lbf
Drehmoment	N·m	lbf·in
Leistung	W	hp
Temperatur	°C	°F

Die Spindeldrehzahl bleibt in min^{-1}. Die Tippschritte werden zu 0,0005, 0,005, 0,05, 0,5 und 2 in, die Tippgeschwindigkeiten zu 4, 12, 60 und 120 in/min.

Die Einheiten ändern, was Studio anzeigt und liest, nicht Ihre Dateien. Ein Programm läuft so, wie es geschrieben ist, in `G20` oder `G21`. Programme, die Studio erzeugt, werden in Millimetern geschrieben (`G21`). Die Tools der KI und die Protokolle arbeiten in Millimetern.

Region: Zahlen, Datum und Geld

Extras > Einstellungen… > Allgemein > Region legt fest, wie Zahlen, Geld und Datumsangaben aussehen: **Wie Windows (Einstellungen > Zeit & Sprache)**, **Wie die Sprache** oder eine Region aus der Liste. Auf derselben Seite:

- **Währung**: ein ISO-4217-Code wie USD, EUR oder CNY; leer bedeutet die Währung der Region;
- **Währungssymbol**: leer bedeutet das eigene Symbol der Währung;
- **Maschinenstundensatz**: Ihre Kosten für eine Stunde Maschinenzeit, für die Kostenschätzungen; 0 zeigt keine an.

Die Region bestimmt das Dezimalzeichen und das Tausendertrennzeichen in den Zahlen von Studio, die Datumsangaben in Läufen, Versionen und Berichten und das Papierformat der Berichte (Letter in den USA und Kanada, sonst A4). Zahlen, die zur Maschine gehören, etwa Positionen und die Zahlen in einem Programm, verwenden immer einen Punkt und kein Tausendertrennzeichen.

Zahlen eingeben

Studio liest eine eingegebene Zahl streng in den Feldern, in denen ein Fehler ein Teil kosten würde: den CAM-Feldern, den Größen und Orten in **Vorbereiten**, den Längen des Messtasters und der Spindeldrehzahl.

- Verwenden Sie ein Dezimalzeichen: einen Punkt oder ein Komma (`13.3` oder `13,3`).
- Schreiben Sie kein Tausendertrennzeichen.
- Sie dürfen die Einheit des Feldes hinter die Zahl schreiben: `13 mm`, `0.5 in`, `30°`.
- Mit einer Eingabemethode getippte Ziffern (Ziffern in Vollbreite) werden ebenfalls gelesen.

`1,000` ist eine Frage, wenn Ihre Region das Dezimalzeichen als Punkt schreibt: Meinten Sie eins oder eintausend? Studio rät nicht; es bittet Sie, die Zahl zu schreiben, die Sie meinen. Wo die Region ein Komma schreibt, ist `1,000` eins.

Was „gelesen als“ bedeutet

Wenn das, was Sie getippt haben, nicht so geschrieben ist, wie Studio Zahlen schreibt, zeigt das Feld, was Studio gelesen hat: Tippen Sie `13,3`, und es zeigt **gelesen als 13.3**. In Zoll zeigt es vier Dezimalstellen: **gelesen als 0.3750**. Prüfen Sie es, bevor Sie weitermachen.

Wenn Studio nicht lesen kann, was Sie getippt haben, wird das Feld rot, und eine Zeile darunter sagt, warum, zum Beispiel „‚1,2,3‘ ist keine Länge“. Studio übernimmt den Wert nicht: Das Feld behält den letzten Wert, den es übernommen hat, bis Sie ihn korrigieren.

Zollbrüche

In Zoll nehmen Längenfelder auch Brüche an:

- `3/8`, `5/4`
- `1 1/2` oder `1-1/2`
- `2 3/8"`, `1/2 in` (das Zollzeichen kann `"`, `”` oder `in` sein)

Das Feld zeigt den Dezimalwert und den Bruch, die es gelesen hat: **gelesen als 0.3750 (3/8")**. Studio übernimmt keinen Dezimalpunkt zusammen mit einem Bruch (`1.5/2`), keinen unechten Bruch nach einer ganzen Zahl (`1 5/4`) und keine Leerzeichen um den Schrägstrich.

Studio zeigt Zollmaße auch als Brüche, bis hinunter zu 1/64 in: Werkzeugdurchmesser und -längen, die Maße des Rohteils, Bohrungsdurchmesser und -tiefen lauten `1/4"` oder `1 1/2"`. Ein Maß, das kein Vielfaches von 1/64 ist, zeigt vier Dezimalstellen, `0.2010 in`. Koordinaten, Vorschübe, Geschwindigkeiten und Messwerte sind immer Dezimalzahlen.

In Millimetern wird ein Bruch nicht gelesen: „Millimeterfelder nehmen keine Brüche an: Schreiben Sie 4,7625 oder wechseln Sie zu Zoll“.

Werkzeugbeschreibungen lesen Zollmaße in beiden Einheitensystemen, zum Beispiel `flat 1/4in n2`.

Spindeldrehzahl

Sie können die Spindeldrehzahl eintippen, statt sie in Schritten von 1.000 min^{-1} zu ändern:

1. Öffnen Sie in **Ausführen** in der rechten Spalte **Handbetrieb**.
2. Doppelklicken Sie auf die Anzeige **Spindeldrehzahl** oder drücken Sie darauf die Eingabetaste.
3. Tippen Sie die Drehzahl ein und drücken Sie die Eingabetaste. Esc oder ein Klick an eine andere Stelle behält die alte Drehzahl bei.

Wenn die Spindel von Hand läuft (kein Programm läuft), geht die neue Drehzahl sofort an die Spindel. Steht sie, merkt sich Studio die Drehzahl für das nächste **Spindel rechts** oder **Spindel links**. Während ein Programm läuft, bestimmt das Programm die Drehzahl, und das Feld öffnet sich nicht.

Eine Drehzahl außerhalb des Bereichs der Spindel färbt das Feld rot und wird nicht übernommen. Der Bereich der MC-1 ist 1.000–24.000 min^{-1}.

Was auf Englisch bleibt

- G-Code und die Dateien, die Studio schreibt und liest (Aufträge, Aufspannungen, Aufzeichnungen, Lizenzdateien);
- die Tools der KI und das, was an KI-Clients und von ihnen gesendet wird;
- die Referenzkapitel, die aus dem Programm selbst erzeugt werden: MCP-Tools, Befehlszeile und Schema der Maschinenprofile;
- die Meldung jedes Prüfprotokolleintrags zu Programmen (das Protokoll hält sie auch in Ihrer Sprache fest);
- in 1.0 in Deutsch, Französisch, Japanisch und Spanisch: die Fehler- und Alarmbeschreibungen der Steuerung, die Hilfe zu den Einstellungen der Steuerung und einige Lizenzmeldungen.

Das PDF eines Prüfberichts ist in Ihrer Sprache, wenn seine Schriften die Buchstaben Ihrer Sprache haben, und sonst auf Englisch; seine CSV-Datei ist immer in Ihrer Sprache.

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Tastatur und Barrierefreiheit

Die Maschine, die Menüs, die Tasten der Panels und der Editor lassen sich alle über die Tastatur bedienen. Das Drehen und Zoomen der 3D-Ansicht, das Wählen eines Punktes darin und das Ändern der Größe von Panels brauchen eine Maus. Unter Windows liest ein Screenreader die Tasten und Anzeigen der Panels.

Hilfe > Tastenkombinationen (oder **?**) listet jede Taste in Studio auf.

Die Tasten für die Maschine

Taste	Funktion
F1	**Not-Halt**: drücken; erneut drücken zum Entriegeln
F2	**Einschalten**
F3	**Referenzfahrt**
F4	**Entsperren**
F5	**Zyklusstart**
F6 oder Leertaste	**Vorschub halt**
F7	**Stopp**: Vorschub halt, danach ein Reset, sobald die Maschine steht; erneut drücken, um sofort zurückzusetzen
F8	**Kühlung** ein oder aus
F9	**Spindel rechts**
F10	**Spindel links**
F11	**Spindelstopp**
F12 oder Strg+X	**Reset**

Tippbetrieb und Handrad:

Taste	Funktion
← →	Tippbetrieb X
↓ ↑	Tippbetrieb Y
Bild↓, Bild↑	Tippbetrieb Z
Umschalttaste mit einer dieser Tasten	Tippbetrieb, solange Sie die Taste halten
1–5	Schritt 0,01, 0,1, 1, 10 oder 50 mm
6	**Gedrückt halten**: Jede Tipptaste fährt, solange Sie sie halten
Strg+1 bis Strg+4	Tippgeschwindigkeit 100, 300, 1.500 oder 3.000 mm/min
Esc	**Tippen stoppen**
X, Y, Z	Die Achse des Handrads wählen
, und .	Das Handrad um eine Raste herunter- oder hinaufdrehen (mit Umschalttaste: zehn)

Overrides:

Taste	Funktion
Strg+↑, Strg+↓	Vorschub +10 %, −10 %
Strg+→, Strg+←	Spindel +10 %, −10 %
Strg+0	Vorschub und Spindel zurück auf 100 %

Jeder Druck auf eine Tipptaste fährt einen Schritt. Ein Tippen bei gehaltener Taste endet, wenn Sie die Taste loslassen oder wenn das Fenster den Fokus verliert. In Zoll betragen die Schritte 0,0005–2 in und die Geschwindigkeiten 4–120 in/min.

Die F-Tasten funktionieren überall. Die übrigen Maschinentasten funktionieren nicht, während Sie in einem Textfeld, einer Liste oder im Editor tippen: Klicken Sie auf die 3D-Ansicht, um sie wieder der Maschine zu übergeben.

Die Zifferntasten und die Tasten , und . gelten nach ihrer Lage auf der Tastatur, deshalb funktionieren sie mit jedem Tastaturlayout.

Not-Halt

Drücken Sie F1 oder klicken Sie auf die runde rote Taste am rechten Ende der Maschinenleiste. Im Zwilling öffnet das den Not-Halt-Kreis wie der Taster der Maschine: Die Antriebe und der Spindelantrieb werden stromlos, und die Steuerung löst ALARM:10 aus. Drücken Sie F1 erneut, um den Not-Halt zu entriegeln, und folgen Sie dann den Schritten zurück zur Arbeit: **Einschalten**, **Reset**, **Referenzfahrt**.

> ⚠ **Warnung**
>
> F1 und die rote Taste stoppen den Zwilling. Sie sind kein Not-Halt für eine reale Maschine. Verwenden Sie an einer realen Maschine den eigenen Not-Halt-Taster der Maschine.

Die Tasten für das Fenster

Taste	Funktion
Alt+1 bis Alt+6	Wechseln zu **Vorbereiten**, **CAM**, **Programm**, **Ausführen**, **Analyse** oder **Diagnose**
Strg+Tab, Strg+Umschalt+Tab	Der nächste oder vorherige Arbeitsbereich
Strg+J	Das Dock ein- oder ausklappen
Strg+Umschalt+M	Maschinen wechseln
Strg+O	**Programm öffnen…**
Strg+P	**Programm drucken…**
Strg+,	**Einstellungen…**
Strg+L	**Prüfprotokoll…**
Strg+Q	**Beenden**
? oder /	**Tastenkombinationen**
Alt und der Buchstabe eines Menüs	Öffnet dieses Menü; Alt allein hebt die Menütitel hervor
Tab, Umschalt+Tab	Zwischen den Tasten eines Panels wechseln; die Leertaste drückt die Taste mit dem Rahmen
Esc	Ein Pop-up oder ein Unterfenster schließen

> **Hinweis**
>
> F1 ist der Not-Halt, nicht die Hilfe. Um das Handbuch zu öffnen, verwenden Sie **Hilfe > Dokumentation** oder die Taste **?** eines Panels.

Der Editor im Arbeitsbereich **Programm** hat eigene Tasten:

Taste	Funktion
Strg+E	In den Editor und zurück zur Maschine
Strg+K	**Laden und prüfen**
Strg+S	**Version speichern**
Strg+Umschalt+S	Das Programm als Datei speichern
Strg+D	**Vergleichen** mit einer gespeicherten Version
Strg+F, Strg+G, Strg+H	Suchen, zu einer Zeile gehen, ersetzen
Strg+B	**Haltepunkt** im Satz
Strg+Umschalt+B	**Anhalten bei** einer Bedingung
Strg+.	Den Vorschlag der Prüfung für den Satz anwenden
Alt+↓, Alt+↑	Der nächste oder vorherige Befund oder Unterschied
Umschalt+F10 oder die Menütaste	Das Menü des Satzes
Strg+Umschalt+O	**Optimieren**
Strg+Umschalt+G	**Generieren…**
Strg+Umschalt+R	Das Programm übernehmen, das in der Maschine ist
Esc	Den Vergleich schließen

Das Fenster des Handbuchs: Strg+F sucht, Alt+← und Alt+→ gehen zurück und vor, Esc schließt es.

Während Sie in **Vorbereiten** ein Werkstück anordnen: Die Pfeiltasten bewegen es (mit der Umschalttaste feiner, mit Strg gröber), Q und E drehen es, die Eingabetaste platziert es, Esc setzt es zurück.

Die vollständige Liste

Hilfe > Tastenkombinationen zeigt alle oben genannten Tasten in einer Liste. Diese Befehle haben keine Taste; verwenden Sie ihre Menüs:

- **Einzelsatz** und **Nächster Satz** (das Menü **Programm** und die Maschinenleiste);
- **Programm prüfen**;

- das Nullsetzen der Achsen und das Anfahren des Nullpunkts (**Maschine > Werkstücknullpunkt**);
- die Bewegungen von Werkzeuglängensensor und Messtaster (**Maschine > Werkzeuglängensensor und Messtaster**);
- die Spindeldrehzahl in Schritten von 1.000 min^{-1} und der Eilgang-Override (**Maschine**);
- das Anhalten der Simulation und ihre Geschwindigkeit (**Simulation**).

Einen Screenreader verwenden

Unter Windows arbeitet Studio über die Barrierefreiheitsschnittstelle des Systems mit Screenreadern zusammen. In jedem Panel von Studio erhält der Screenreader:

- den Namen jeder Taste, welche Art von Taste sie ist, ob sie eingeschaltet, geöffnet oder nicht verfügbar ist, ihre Tastenkombination und ihren Tooltip;
- die Anzeigen als Text: die Maschinenleiste (Zustand, Programm, Fortschritt, der nächste Schritt, die Uhr), die obere Leiste (Maschine, Zustand, KI, Stufe, Edition) und die rechte Spalte (Position, Spindel, Instrumente, die Zeile der modalen Codes);
- jede Bewegung des Fokus mit Tab;
- die Menüs: den Text jedes Eintrags, seine Tastenkombination, ob er nicht verfügbar oder angehakt ist, und bei welchem Eintrag Sie sind.

Zwei Dinge werden sofort mitgeteilt, wo auch immer der Fokus ist: ein Alarm (mit dem nächsten Schritt) und die Anfrage einer KI, die Maschine zu ändern.

Das Fenster des Handbuchs gibt dem Screenreader den Text der Seite, mit ihren Überschriften und Links.

In 1.0:

- andere Änderungen des Zustands der Maschine werden nicht mitgeteilt: Setzen Sie den Fokus auf die Maschinenleiste, um sie zu hören;
- die Programmliste, die Konsole, die Meldungen, der Editor und die Werkzeugtabelle werden nach den eigenen Regeln des Screenreaders gelesen; die Kurven werden nicht gelesen;
- unter Linux werden Screenreader nicht unterstützt.

Was der Screenreader in der 3D-Ansicht liest

Die 3D-Ansicht selbst gibt dem Screenreader nichts. Die Zahlen, die sie zeigt, stehen in der rechten Spalte und in der Maschinenleiste, die der Screenreader liest. Wenn Sie auf einen Punkt des Werkstücks klicken, wird der Text unter der Ansicht (der Programmsatz und die Position) gelesen.

Große Schrift und Werkstattmodus

Studio folgt der Anzeigeskalierung, die Sie in Windows einstellen, Bildschirm für Bildschirm. Es hat keine eigene Einstellung für die Textgröße.

Ansicht > Darstellung > Werkstattmodus (groß) vergrößert die Maschinenleiste, ihre Tasten, den Not-Halt, den Text, die Programmliste und die Position und gibt dem Arbeitsbereich **Ausführen** ein Layout zum Lesen auf Armlänge (siehe Hell, Dunkel und Werkstattmodus).

Wenn die Animationen des Systems ausgeschaltet sind, springt die 3D-Ansicht zu einer neuen Ansicht, statt dorthin zu gleiten.

Fehlerbehebung

Was Sie tun können, wenn etwas nicht so funktioniert, wie Sie es erwarten. Jeder Abschnitt sagt, was Sie sehen, was es meist bedeutet und wo die vollständige Antwort steht. Wenn keiner passt, sammeln Sie die Diagnoseinformationen und schreiben Sie uns (der letzte Abschnitt).

Studio startet nicht

- Studio braucht 64-Bit-Windows 10 oder 11; auf anderen Versionen bricht das Installationsprogramm ab. Siehe Systemvoraussetzungen.
- Wenn sich das Fenster öffnet, die 3D-Ansicht aber leer bleibt oder meldet, dass sie nicht verfügbar ist: Setzen Sie unter **Extras > Einstellungen… > Dateien** das Häkchen bei **Software-Rendering (bei Problemen mit der Grafikkarte; ab dem nächsten Start)** und starten Sie Studio erneut. Oft behebt ein Update des Grafikkartentreibers die Ursache. Siehe Wenn die 3D-Ansicht nicht starten kann.
- Wenn sich Studio beim Starten geschlossen hat, bietet der nächste Start den Bericht an (siehe Studio hat sich unerwartet geschlossen).

Die Maschine ist im Alarm

Im Alarm wird der Zustand in der Maschinenleiste rot. Die Leiste nennt den Alarm und was er bedeutet und gibt den **nächsten Schritt** an, mit der Taste, die ihn ausführt; **Alle Schritte** listet den Weg zurück zur Arbeit auf, wenn es mehr als einen gibt. Klicken Sie auf die Taste oder verwenden Sie **Maschine > Nächsten Schritt ausführen**: Studio berechnet die Schritte nach jedem Schritt neu.

- Eine Antastbewegung, bei der nichts berührt wurde, stoppt mit ALARM:5: Führen Sie die Schritte aus (**Reset**, **Entsperren**) und prüfen Sie dann, wo der Zyklus die Fläche erwartet hat.
- Eine Bewegung über den Verfahrbereich der Maschine hinaus stoppt am Software-Endschalter: Sehen Sie sich den Satz an, den die Programmliste rot markiert, und den Werkstücknullpunkt, mit dem er lief.
- Nach einem Alarm ist die Position möglicherweise nicht mehr verlässlich (eine Zeile unter **Position** sagt das): Referenzieren Sie erneut, bevor Sie fräsen.

Die Schritte für jeden Fall: Alarme und was zu tun ist. Jeder Alarm, seine Ursachen und was ihn löscht: Alarme.

Ein Satz des Programms antwortet mit einem Fehler

`error:` und eine Zahl bedeuten, dass die Steuerung diesen Satz abgelehnt hat: Sie hat ihn nicht ausgeführt, und die Maschine hat sich dafür nicht bewegt. Die Meldung nennt die Zahl und was sie bedeutet.

- **Programm > Programm prüfen** schickt das ganze Programm durch den Prüfmodus der Steuerung, während sich die Maschine nicht bewegt, und listet die ersten Sätze auf, die sie abgelehnt hat: „Satz 5: error:22 (…)“. Korrigieren Sie diese Sätze und prüfen Sie erneut.
- Die häufigsten Ursachen sind ein Code, den die Steuerung nicht annimmt, ein fehlendes Wort (ein Kreisbogen ohne Mittelpunkt oder Radius, eine Schnittbewegung vor dem ersten `F`) oder zwei Codes in einem Satz, die nicht zusammenpassen.

Jeder Fehler und seine Ursachen: Fehler.

Die Programmprüfung findet ein Problem

Der Editor prüft das Programm, während Sie es eingeben, beim Laden und nach einem Lauf. Jeder Befund steht unter seinem Satz mit der Nummer seiner Regel und seiner Art: ein Fehler, eine Warnung oder ein Hinweis.

- Ein **Fehler** ist ein Satz, den die Steuerung ablehnen würde, oder eine Bewegung, die die Aufspannung nicht zulässt; beheben Sie ihn, bevor Sie ausführen.
- Eine **Warnung** ist ein Satz, der läuft, aber möglicherweise nicht tut, was Sie meinen: ein Eilgang unterhalb der Oberkante des Werkstücks, eine ohne `S` gestartete Spindel, ein Werkzeugwechsel, nach dem das Programm nach unten fährt, bevor es über seinem Startpunkt steht.
- Der Vorschauteil der Prüfung braucht die eingeschaltete und referenzierte Maschine und kein laufendes Programm; bis dahin wartet er.

Die Regeln und wie Sie einen Befund beheben: Die Programmprüfung. Was ein Lauf im Zwilling darüber hinaus prüft: Was der Zwilling prüft.

Der Antastzyklus startet nicht

Antasten starten… bleibt grau, solange eine Prüfung auf der Messtasterkarte rot ist oder etwas fehlt; die Zeile unter den Tasten sagt dann, was (X und Y angeben, den Durchmesser des Rings angeben, auf einen laufenden Aufruf warten).

- **In der Spindel ist kein Messtaster**: Wählen Sie in **Vorbereiten** im Schritt **Werkzeug** den Messtaster und klicken Sie auf **Werkzeug wechseln**.
- **Die Spindel dreht sich**: Halten Sie sie an (**Spindelstopp**, F11).
- **Kein Platz über dem Rohteil, um darüber hinwegzufahren**: Die Zeile nennt die Höhe der Messtasterspitze bei Z oben, die Oberkante der Aufspannung und wie viel fehlt. Verwenden Sie einen kürzeren Messtaster oder spannen Sie das Rohteil tiefer: Spannpratzen oder doppelseitiges Klebeband statt des Schraubstocks.
- **Der Zwilling folgt einer realen Maschine**: Tasten Sie dort mit der Steuerung der Maschine selbst an.
- Ein Zyklus, der nichts berührt hat, stoppt mit ALARM:5 (siehe Die Maschine ist im Alarm).

Siehe Wenn ein Zyklus stoppt.

Lizenz und Aktivierung

Hilfe > Lizenz zeigt die gültige Edition, den Zustand der Lizenz und wann sie endet.

- **Online aktivieren**: Geben Sie den Code so ein, wie Sie ihn erhalten haben; **Aktivieren** ist verfügbar, sobald der Code als Code gelesen wird. Die Antwort des Servers erscheint darunter. Lautet die Antwort, dass die Lizenz auf zu vielen Computern ist, abgelaufen ist, pausiert ist, nicht bezahlt oder erstattet wurde, öffnet **Lizenzseite öffnen** sie auf unserer Website: Dort geben Sie einen Computer frei, verlängern oder verwalten das Abonnement.
- **Ohne Internet**: Derselbe Dialog enthält die Schritte für den Offline-Betrieb. Studio gibt einen Anforderungscode aus; auf einem Computer mit Internet erhalten Sie dafür auf der Lizenzseite die Lizenzdatei und importieren diese Datei dann hier.
- Ein Abonnement, das nicht verlängert wird, funktioniert noch 14 Tage weiter (die Lizenz zeigt **In Kulanzfrist**). Danach oder wenn eine Lizenz endet, ist Studio wieder die Learning-Edition; Ihre Projekte und Programme bleiben unberührt.

Siehe Lizenzierung und Wenn eine Lizenz endet.

Der KI-Assistent antwortet nicht

- Die Zeile unter der Eingabe sagt, was der Assistent gerade tut (auf den Dienst warten, der Zwilling führt ein Tool aus).
- Prüfen Sie den **Anbieter**, die **Basis-URL** und das **Modell**; bei einem Cloud-Dienst, dass ein Schlüssel gespeichert ist (**Schlüssel speichern**).
- Bei einem lokalen Modell muss dessen Server laufen (der Server von LM Studio, Ollama) und ein Modell gewählt sein (**Modelle auflisten**).
- Eine Antwort kann eine Weile dauern; **Stopp** beendet sie.
- Die erste Nachricht an einen Cloud-Dienst wartet auf Sie: Studio zeigt einmal, was dorthin geht, und sendet erst, wenn Sie zustimmen (für jeden Dienst einmal gefragt). **Lokales Modell verwenden…** behält alles auf diesem Computer. Siehe Datenschutz.
- Wenn die KI die Maschine im Fenster ändern will, fragt Sie eine Leiste am unteren Rand des Fensters. Ohne Antwort innerhalb von 60 s gilt die Anfrage als abgelehnt.

Siehe Der KI-Assistent.

Studio hat sich unerwartet geschlossen

Beim nächsten Start fragt Studio, was mit dem Bericht über das Geschehene gemacht werden soll: seinen Ordner öffnen oder ihn als Datei speichern. Es wird nichts gesendet. Der Bericht lässt Programme, Datei- und Ordnernamen, Schlüssel, Codes und Namen von Personen weg; Berichte, die älter als 30 Tage sind, werden beim Start gelöscht. Siehe Datenschutz.

Hilfe erhalten

Hilfe > Problem melden… sendet ein Problem, eine Frage zur Bedienung oder eine Idee an den Support von Twinwright. Schreiben Sie, was Sie getan, was Sie erwartet und was Sie gesehen haben; geben Sie eine E-Mail-Adresse an, wenn Sie die Antworten auch dort erhalten möchten. Immer mitgesendet und im Dialog angezeigt (**Alle anzeigen…** zeigt es vollständig): die Version und der Build, die Edition, das System mit seiner Sprache und seinem Bildschirm sowie die letzten 50 Einträge des Prüfprotokolls in Kurzform – was getan wurde und wie es ausging, keine Programminhalte, Ihr eigener Ordner als `~` geschrieben. **Diagnosepaket anhängen** und **Screenshot des Fensters anhängen** haben beim Öffnen des Dialogs kein Häkchen; davon wird nichts gesendet, außer Sie setzen das Häkchen. Nach **Senden** zeigt der Dialog die Nummer des Tickets (zum Beispiel `TW-1042`). Ohne Verbindung wird die Meldung als Datei gespeichert, `problem-<time>.zip`, in `Dokumente\Twinwright Studio\Reports`: Senden Sie sie erneut aus dem Dialog, oder laden Sie sie auf der Support-Seite der Website hoch. **Als Datei speichern…** speichert sie, ohne sie zu senden.

Ein KI-Client kann eine Meldung für Sie entwerfen (report_draft): Derselbe Dialog öffnet sich mit seinem Text und einer Zeile, die angibt, welcher Client ihn geschrieben hat. Lesen Sie ihn und ändern Sie, was Sie möchten; nur Sie können die Meldung senden, und nur Sie können das Paket oder den Screenshot anhängen.

Hilfe > Meine Meldungen listet die Meldungen auf, die von diesem Computer gesendet wurden: die Nummer jeder Meldung, ihren Status (eingegangen, in Bearbeitung, beantwortet, gelöst) und wann sie sich zuletzt geändert hat; ein Punkt kennzeichnet eine Antwort, die Sie noch nicht gelesen haben. Wählen Sie eine aus, um den Gesprächsverlauf zu lesen; **Etwas ergänzen…** sendet weitere Angaben zum selben Ticket (**Diagnosepaket anhängen…** fügt ihm das Paket hinzu). Die Antworten stehen auch in Ihrem Kundenkonto unter „Meine Tickets“. Solange eine Meldung nicht gelöst ist, fragt Studio alle 30 Minuten nach neuen Antworten, und das Menü Hilfe zeigt, wie viele neu sind; **Einstellungen > Allgemein > Alle 30 Minuten nach Antworten auf meine Meldungen suchen** schaltet das ab. Was gesendet und gespeichert wird: Datenschutz.

Hilfe > Diagnoseinformationen sammeln… legt alles, was der Support braucht, in einer ZIP-Datei in `Dokumente\Twinwright Studio\Reports` ab: die Version, die Einstellungen (Schlüssel als `***` geschrieben), das Maschinenarchiv, die Einstellungen der Steuerung, die letzten 7 Tage des Prüfprotokolls, die Absturzberichte und den Zustand der Lizenz. **Programm und Aufspannung jetzt** wird weggelassen, außer Sie setzen das Häkchen. Pfade, Dateinamen und der Benutzername des Computers werden entfernt.

Öffnen Sie die Datei und sehen Sie sie an, bevor Sie sie senden; diese Datei wird nicht von selbst gesendet (eine Meldung nimmt sie nur mit, wenn Sie das Häkchen bei **Diagnosepaket anhängen** setzen). Die Adresse, an die Sie schreiben, steht unter **Hilfe > Info**.

Operations manual

Systemvoraussetzungen

Was ein Computer braucht, damit Twinwright Studio läuft, und was die 3D-Ansicht von der Grafik verlangt.

Windows

- 64-Bit-Windows 10 oder Windows 11 (x64). Das Installationsprogramm bricht bei anderen Versionen ab.
- Sonst muss nichts installiert werden: Studio ist eine einzige Programmdatei und nutzt nur die Bibliotheken, die zu Windows gehören.

Linux

- Linux auf x86-64 mit glibc 2.38 oder neuer: Ubuntu 24.04, Debian 13, Fedora 39 und neuer sowie die darauf aufbauenden Distributionen.
- Ein Desktop mit X11 oder Wayland.
- Für das AppImage: FUSE 2 (unter Ubuntu das Paket `libfuse2t64`). Ohne FUSE 2 nehmen Sie das `.tar.gz`.
- Für Online-Aktivierung und Testphasen: libcurl (unter Ubuntu das Paket `libcurl4t64`; Desktops haben es meist). Ohne es aktiviert Studio offline.
- Für die Schlüssel der KI-Dienste: der Schlüsselbund des Desktops (Secret Service: GNOME Keyring oder KWallet). Ohne ihn bewahrt Studio die Schlüssel in einer Datei auf, die nur Sie lesen können.

Grafik

Die 3D-Ansicht nutzt Direct3D 11 auf jeder Grafikkarte oder integrierten Grafik, die es unterstützt (Feature Level 10.0 oder höher). Ohne eine solche Grafik zeichnet Studio mit dem Software-Renderer von Windows (WARP): Das funktioniert, aber die Ansicht ist langsamer (höchstens 30 Bilder pro Sekunde), und es gibt keine Werkzeug-Nahansicht.

Hilfe > Info zeigt unter **Anzeige**, welche davon Studio verwendet.

Wenn die 3D-Ansicht nicht starten kann

Der Rest des Fensters funktioniert auch ohne die 3D-Ansicht. Wenn die Ansicht leer bleibt oder anzeigt, dass sie nicht verfügbar ist:

1. Öffnen Sie **Extras > Einstellungen… > Dateien**.
2. Setzen Sie ein Häkchen bei **Software-Rendering (bei Problemen mit der Grafikkarte; ab dem nächsten Start)**.
3. Schließen Sie Studio und starten Sie es erneut.

Oft behebt die Aktualisierung des Grafikkartentreibers die Ursache.

Linux

Unter Linux verwendet die 3D-Ansicht OpenGL 4.1 oder höher: Die Mesa-Treiber haben es für die Grafik der letzten zehn Jahre. Ohne Grafiktreiber zeichnet Mesas Software-Renderer die Ansicht, langsamer; um ihn zu probieren, starten Sie Studio aus einem Terminal mit `LIBGL_ALWAYS_SOFTWARE=1` vor dem Programmnamen.

Arbeitsspeicher und Festplattenspeicher

- Arbeitsspeicher: 8 GB oder mehr.
- Festplattenspeicher: etwa 62 MB für das Programm und sein Handbuch sowie Platz für Ihre Daten. Ihre Daten wachsen mit der Nutzung: Das Prüfprotokoll wird standardmäßig 180 Tage lang und bis zu 2 GB aufbewahrt (**Extras > Einstellungen… > Protokoll**), und jede Programmversion und jeder Bericht bleibt erhalten (siehe Ihre Daten und Backups).

Bildschirme und Skalierung

- Mindestens 1280 × 720; Studio ist auf 1600 × 1000 und mehr ausgelegt.
- Jede Anzeigeskalierung. Studio folgt der Skalierung des jeweiligen Bildschirms und skaliert das Fenster neu, wenn Sie es auf einen anderen Bildschirm verschieben.

Eine echte Maschine

In 1.0 ist das Verbinden mit einer echten Maschine eine Vorschau und erfolgt über die Befehlszeile (siehe Eine echte Maschine): über einen seriellen Port (USB) mit 115.200 Baud oder über eine Steuerung im Netzwerk.

Studio nutzt das Netzwerk wie folgt:

- G-Code-Sender-Programme verbinden sich mit dem Zwilling auf diesem Computer, Port 2323;
- die Browser-Ansicht nutzt Port 8765 auf diesem Computer;
- KI-Clients, die eine Adresse annehmen, nutzen Port 8766 auf diesem Computer, wenn Sie ihn einschalten;
- die Online-Aktivierung und -Verlängerung gehen an den Lizenzserver.

Die Ports von Studio nehmen nur Verbindungen von diesem Computer an: Andere Computer können sie nicht erreichen.

Unter Linux gehört die serielle Schnittstelle der Gruppe `dialout`: Fügen Sie Ihr Konto einmal hinzu (`sudo usermod -aG dialout $USER`) und melden Sie sich neu an.

Installieren und Entfernen

Wie Sie Twinwright Studio installieren, auch ohne Rückfragen auf vielen Computern gleichzeitig, was wohin kommt und wie Sie es wieder entfernen.

Für Linux siehe Unter Linux.

Installieren

1. Starten Sie `TwinwrightStudio-<version>-setup.exe`.
2. Lesen Sie die Lizenzvereinbarung und klicken Sie auf **I Agree**.
3. Behalten Sie den Ordner (`%LOCALAPPDATA%\Programs\Twinwright Studio`) bei oder wählen Sie einen anderen.
4. Wählen Sie auf der letzten Seite, ob Studio gestartet werden soll und ob eine Verknüpfung auf dem Desktop angelegt werden soll.

Das Installationsprogramm erscheint auf Englisch oder Chinesisch, passend zur Sprache des Systems.

Für Sie oder für alle Benutzer des Computers

Das Installationsprogramm installiert Studio für die Person, die es ausführt. Es braucht keine Administratorrechte, und der Eintrag im Startmenü und die Verknüpfung auf dem Desktop gehören dieser Person. Jede Person, die den Computer nutzt, installiert Studio für sich selbst.

Die Lizenz gehört dem Computer, nicht der Person: Alle Benutzer des Computers teilen sie (siehe Der Lizenzordner).

Studio ändert weder Dateizuordnungen noch `PATH` noch Firewall-Einstellungen und installiert keinen Dienst.

Installieren ohne Rückfragen

```
TwinwrightStudio-<version>-setup.exe /S [/DESKTOP] [/D=<folder>]
```

- `/S` installiert ohne Dialogseiten und startet Studio nicht.
- `/DESKTOP` legt die Verknüpfung auf dem Desktop an.
- `/D=` legt den Ordner fest. Diese Angabe muss das letzte Argument sein, ohne Anführungszeichen, auch wenn der Ordner Leerzeichen enthält.

Um Studio ohne Rückfragen zu entfernen: `uninstall.exe /S` im Ordner des Programms.

Was wohin kommt

Was	Wo
Das Programm	`%LOCALAPPDATA%\Programs\Twinwright Studio`
Ihre Arbeit: Programme, Versionen, Berichte, das Prüfprotokoll	`Dokumente\Twinwright Studio`
Der eigene Zustand des Programms: Einstellungen, Maschinen, KI-Schlüssel	`%LOCALAPPDATA%\Twinwright\Studio`
Die Lizenz	`%ProgramData%\Twinwright\licence`

Der Ordner des Programms

`TwinwrightStudio.exe`, das Handbuch (`manual\`), `README.txt`, `THIRD-PARTY-NOTICES.txt` und `uninstall.exe`. Der Ordner bleibt bei Updates derselbe, sodass KI-Clients, die Sie verbunden haben, weiter funktionieren.

Ihre Dokumente

`Dokumente\Twinwright Studio` (der Ordner „Dokumente“, den Windows Ihnen zuweist, auch wenn er umgeleitet ist):

- `programs\`: die Versionen jedes Programms und die Versionen der CAM-Aufträge;
- `Reports\`: Prüfberichte, Einrichteblätter und Diagnosepakete;
- `logs\`: das Prüfprotokoll.

Mit **Extras > Einstellungen… > Dateien** und **Protokoll** lassen sich die Programme und das Prüfprotokoll an einen anderen Ort verlegen (ab dem nächsten Start).

Der eigene Zustand des Programms

`%LOCALAPPDATA%\Twinwright\Studio`: die Einstellungen, die Position des Fensters, die Welt jeder Maschine (ihr Werkstück, ihre Werkzeuge, ihre Nullpunktverschiebungen und ihre Steuerungseinstellungen), die Schlüssel der KI-Dienste (für Ihr Windows-Konto verschlüsselt) und Absturzberichte (30 Tage lang aufbewahrt).

Der Lizenzordner

`%ProgramData%\Twinwright\licence` enthält die Lizenzdateien, Testphasen eingeschlossen, für alle Benutzer des Computers. Studio legt ihn bei der ersten Aktivierung oder Testphase an. Kann Studio dort nicht schreiben, kann ein Administrator den Ordner anlegen und Benutzern erlauben, darin zu schreiben.

Für besondere Konfigurationen verschieben diese Umgebungsvariablen die Ordner:

Variable	Verschiebt
`TWINWRIGHT_STATE_DIR`	Den eigenen Zustand des Programms
`TWINWRIGHT_WORKDIR`	Den Ordner mit den Programmen
`TWINWRIGHT_LOGDIR`	Das Prüfprotokoll
`TWINWRIGHT_LICENCE_DIR`	Den Lizenzordner

Auf eine neue Version aktualisieren

Schließen Sie Studio und führen Sie dann das Installationsprogramm der neuen Version aus. Es installiert in denselben Ordner; Ihre Arbeit, Ihre Einstellungen, Maschinen und Lizenz bleiben unverändert. Studio sucht nicht selbst nach Updates (siehe Updates).

Läuft im Hintergrund noch eine Kopie von Studio (ein KI-Client kann eine gestartet haben), verschiebt das Installationsprogramm die alte Programmdatei beiseite und entfernt sie beim nächsten Update oder Entfernen.

Entfernen

Verwenden Sie in Windows **Einstellungen > Apps > Twinwright Studio > Deinstallieren** oder führen Sie `uninstall.exe` im Ordner des Programms aus.

Was beim Entfernen übrig bleibt

Das Entfernen löscht das Programm, seine Verknüpfungen und seinen Eintrag unter **Apps**. Es bleiben erhalten:

- Ihre Arbeit in `Dokumente\Twinwright Studio`;
- der eigene Zustand des Programms in `%LOCALAPPDATA%\Twinwright\Studio`;
- der Lizenzordner;
- der Eintrag `twinwright` in den Einstellungen der KI-Clients, die Sie verbunden haben.

Löschen Sie diese von Hand, um alles zu entfernen. Das Löschen des Lizenzordners gibt keine Testphase zurück: Der Lizenzserver gibt jedem Computer von jeder Edition eine Testphase.

Unter Linux

Installieren

Studio für Linux ist eine einzige Datei, ein AppImage. Im System wird nichts installiert.

1. Prüfen Sie die Datei: `sha256sum TwinwrightStudio-<version>-x86_64.AppImage` muss die SHA-256 von der Download-Seite ergeben.
2. Legen Sie sie an einen festen Ort, unter einem Namen ohne Version, damit verbundene KI-Clients nach einem Update weiter funktionieren, und machen Sie sie ausführbar:

```
mkdir -p ~/Applications
mv TwinwrightStudio-<version>-x86_64.AppImage ~/Applications/TwinwrightStudio.AppImage
chmod +x ~/Applications/TwinwrightStudio.AppImage
```

1. Starten Sie es: `~/Applications/TwinwrightStudio.AppImage`, oder doppelklicken Sie es im Dateimanager.

Das AppImage braucht FUSE 2. Startet es deshalb nicht, installieren Sie es (`sudo apt install libfuse2t64` unter Ubuntu 24.04 und neuer), oder nehmen Sie stattdessen das `.tar.gz`: irgendwo entpacken und `TwinwrightStudio` in seinem Ordner starten.

Was wo liegt unter Linux

Was	Wo
Das Programm, mit seinem Handbuch	das AppImage, wo Sie es abgelegt haben
Ihre Arbeit: Programme, Versionen, Berichte, das Audit-Protokoll	`~/Documents/Twinwright Studio`
Der eigene Zustand des Programms: Einstellungen, Maschinen	`~/.local/state/twinwright/studio` (unter `$XDG_STATE_HOME`, wenn gesetzt)
Die Schlüssel der KI-Dienste	der Schlüsselbund des Desktops; ohne ihn eine Datei im Zustandsordner, die nur Sie lesen können
Die Lizenz	`/etc/twinwright/licence` oder `~/.local/share/Twinwright/licence`

Der Lizenzordner gehört dem Computer, für alle Benutzer, wenn Ihr Konto ihn anlegen oder darin schreiben kann. Sonst bewahrt Studio die Lizenz in `~/.local/share/Twinwright/licence` auf (unter `$XDG_DATA_HOME`, wenn gesetzt): nur für Sie, und weiterhin nur für diesen Computer. Um eine Lizenz unter den Benutzern eines Computers zu teilen, legen Sie den Ordner einmal an mit `sudo install -d -m 1777 /etc/twinwright/licence`.

Testphase und Aktivierung unter Linux

Testphasen, Aktivierung und Verlängerung laufen online wie unter Windows, über das libcurl des Systems. Ohne libcurl oder ohne Internet aktivieren Sie offline (siehe Lizenzen).

Aktualisieren und entfernen unter Linux

Zum Aktualisieren beenden Sie Studio und legen das neue AppImage an die Stelle des alten, unter demselben Namen (Schritt 2 oben). Zum Entfernen löschen Sie das AppImage. Beides lässt Ihre Dokumente, den Zustandsordner, den Lizenzordner, die Schlüssel in Ihrem Schlüsselbund und den Eintrag `twinwright` in verbundenen KI-Clients zurück; löschen Sie diese von Hand, um alles zu entfernen.

Lizenzen, Arbeitsplätze und Aktivierung

Ohne Lizenz läuft Studio als Edition Learning. Eine kostenpflichtige Lizenz fügt die Funktionen ihrer Edition hinzu. Dieses Kapitel richtet sich an die Person, die Lizenzen kauft, aktiviert und verwaltet. Die Preise stehen auf der Website.

Falls kostenpflichtige Lizenzen noch nicht erhältlich sind, wenn Sie dies lesen, steht das auf der Preisseite der Website. Bis dahin läuft Studio als Edition Learning, und Sie können Maker oder Pro 30 Tage kostenlos testen (siehe Eine kostenpflichtige Edition testen); die Abschnitte zur Aktivierung gelten, sobald Lizenzen verkauft werden.

Editionen

Edition	Fügt hinzu	Wer sie nutzen darf
Learning	Den ganzen Zwilling, den Editor, die reale Maschine, die Beispiele, die KI-Anbindung und CAM für Taschen, Konturen, Bohren, Gravieren und 3D-Parallelschlichten	Alle, kostenlos, zum Lernen, Unterrichten und für private Projekte, mit denen kein Geld verdient wird
Maker	Adaptives Ausräumen, alle 3D-Schlichtstrategien, Restmaterialbearbeitung, zwei Seiten und mehrere Aufspannungen	Eine Person, auch zum Verkauf dessen, was sie herstellt
Pro	Alles aus Maker, dazu Verschachtelung, den Programmoptimierer und das Prüfen mit dem Messtaster (Abgleich mit der Konstruktion) mit seinen Berichten	Ihre Organisation, für jede gewerbliche Nutzung, je Arbeitsplatz

Eine Lizenz **MC-1-Besitzer** ist Pro für einen Arbeitsplatz, zur Nutzung mit der MC-1, deren Seriennummer sie nennt. Sie geht mit der Maschine auf den nächsten Besitzer über.

Eine kostenpflichtige Lizenz gibt es in diesen Arten:

- **Abonnement**: monatlich oder jährlich, bis Sie es kündigen.
- **Kauflizenz**: gehört Ihnen dauerhaft. Sie gilt für alle Versionen, die innerhalb von 12 Monaten nach dem Kauf oder nach einer bezahlten Verlängerung der Updates erscheinen.
- **MC-1-Besitzer**: wie eine Kauflizenz, an die Maschine gebunden.

Arbeitsplätze und Computer

Ein kostenpflichtiger Arbeitsplatz wird jeweils von einer Person genutzt, auf bis zu drei Computern. Ein Computer, den mehrere Personen gemeinsam nutzen, zählt einmal.

Jede Aktivierung bindet die Lizenz über einen Hash des Fingerabdrucks an einen Computer. Der Fingerabdruck selbst verlässt den Computer nie. Welche Computer eine Lizenz nutzen, sehen Sie auf der Lizenzseite twinwright.com/de/licence, nachdem Sie den Aktivierungscode eingegeben haben.

Eine kostenpflichtige Edition testen

Jede kostenpflichtige Edition können Sie 30 Tage lang kostenlos testen, je einmal auf jedem Computer.

1. Nutzen Sie eine kostenpflichtige Funktion, zum Beispiel das adaptive Ausräumen im CAM-Bereich. Ergebnis: Studio nennt die Edition, die die Funktion braucht.
2. Wählen Sie **30 Tage testen…**.
3. Lesen Sie, was die Testphase hinzufügt und bis wann, und wählen Sie dann **Testphase starten**. Ergebnis: Studio fragt beim Lizenzserver von Twinwright die Testphase an, und die Funktionen der Edition stehen bis zum angezeigten Datum zur Verfügung.

Für eine Testphase sendet Studio die Edition, einen Hash des Fingerabdrucks des Computers und die Studio-Version, sonst nichts. Der Server gibt jedem Computer von jeder Edition eine Testphase: Installieren Sie Studio während der Testphase neu, bringt ein erneuter Start der Testphase dasselbe Enddatum zurück. Wenn die Testphase endet, ist Studio wieder die Edition Learning; Ihre Projekte und Programme bleiben unverändert.

Auf einem Computer ohne Internet zeigt Studio stattdessen einen Anforderungscode an. Öffnen Sie auf einem Computer mit Internet die Lizenzseite, twinwright.com/de/licence, wählen Sie unter „Auf einem Computer ohne Internet testen" die Edition, fügen Sie den Anforderungscode ein und laden Sie die Testdatei herunter. Importieren Sie die Datei dann auf dem ersten Computer, wie unter Einen Computer ohne Internet aktivieren.

Online aktivieren

Sie brauchen den Aktivierungscode aus der E-Mail zu Ihrer Lizenz, in der Form `TW-XXXXX-XXXXX-XXXXX-XXXXX`, und eine Internetverbindung.

1. Wählen Sie **Hilfe > Lizenz…**.
2. Wählen Sie **Online aktivieren…**.
3. Geben Sie den Aktivierungscode ein. Groß- und Kleinschreibung, Leerzeichen und Bindestriche spielen keine Rolle.
4. Wählen Sie **Aktivieren**. Ergebnis: Die Lizenzdatei liegt im Lizenzordner, und der Dialog zeigt die Edition.

Beim Aktivieren werden nur der Aktivierungscode, ein Hash des Fingerabdrucks dieses Computers und die Studio-Version gesendet.

Läuft die Lizenz schon auf so vielen Computern, wie sie erlaubt, geben Sie zuerst auf der Lizenzseite einen frei (siehe Eine Lizenz auf einen anderen Computer übertragen).

Einen Computer ohne Internet aktivieren

So aktivieren Sie einen Labor- oder Werkstattcomputer, der nie online ist. Sie brauchen den Aktivierungscode, einen zweiten Computer mit Internet und eine Möglichkeit, eine kleine Datei zu übertragen, zum Beispiel einen USB-Stick.

1. Wählen Sie auf dem Computer ohne Internet **Hilfe > Lizenz…**.
2. Wählen Sie **Offline aktivieren…**.
3. Wählen Sie **Kopieren**. Ergebnis: Der Anforderungscode ist in der Zwischenablage. Er enthält einen Hash des Fingerabdrucks dieses Computers und die Studio-Version, keine persönlichen Daten.
4. Fügen Sie den Anforderungscode in eine Textdatei ein und bringen Sie sie zum Computer mit Internet.
5. Öffnen Sie dort twinwright.com/de/licence. Fügen Sie unter „Ein Computer ohne Internet" den Aktivierungscode und den Anforderungscode ein.
6. Wählen Sie „Lizenzdatei herunterladen". Ergebnis: Eine Datei mit einem Namen wie `twinwright-TWL-….twlic` wird heruntergeladen.
7. Bringen Sie die Datei zurück. Wählen Sie in Studio **Hilfe > Lizenz…** und **Lizenzdatei importieren…** und öffnen Sie die Datei. Ergebnis: Die Lizenzdatei liegt im Lizenzordner.

Ein Computer ohne Internet verlängert seine Lizenz nicht von selbst. Wiederholen Sie diese Schritte:

- vor dem Ende eines Abonnementzeitraums;
- bei einer Kauflizenz einmal nach den ersten 30 Tagen (dem Erstattungszeitraum), um die dauerhafte Datei zu erhalten;
- nachdem Sie weitere Jahre Updates gekauft haben.

Der Lizenzordner

Der Lizenzordner gehört zum Computer, nicht zu einem Windows-Konto. Eine Lizenz darin gilt für alle, die den Computer nutzen.

System	Ordner
Windows	`%ProgramData%\Twinwright\licence`
Linux	`/etc/twinwright/licence`; wo er nicht beschreibbar ist: `~/.local/share/Twinwright/licence`

Die Umgebungsvariable `TWINWRIGHT_LICENCE_DIR` legt einen anderen Ordner fest. Um den Ordner zu öffnen, wählen Sie **Hilfe > Lizenz…** und **Lizenzordner öffnen**.

Der Ordner enthält die Lizenzdateien (`*.twlic`, auch die der Testphasen) und eine Datei mit dem Zeitpunkt, zu dem Studio zuletzt nachgesehen hat. Er kann mehrere Lizenzdateien enthalten, zum Beispiel eine MC-1-Besitzerlizenz und ein Abonnement. Es gilt die höchste Edition aller gültigen Lizenzdateien.

> **Hinweis**
>
> Bearbeiten Sie die Dateien nicht. Eine geänderte Lizenzdatei zählt nicht mehr: Laden Sie sie auf der Lizenzseite erneut herunter und importieren Sie sie.

Kann Studio den Ordner nicht anlegen oder nicht hineinschreiben, legt ihn ein Administrator an und gibt den Benutzern Schreibrechte. Unter Linux importieren Sie als root (`sudo`) oder setzen `TWINWRIGHT_LICENCE_DIR`.

Verlängerung und die 14 Tage Kulanzfrist

Ist **Vor dem Ablauf verlängern (gesendet werden nur die Lizenznummer, ein Hash des Fingerabdrucks dieses Computers und die Studio-Version)** eingeschaltet, verlängert Studio die Lizenz selbst, sobald es online ist:

- ein Abonnement ab fünf Tagen vor dem Ende seines Zeitraums;
- eine Kauflizenz am Ende des Erstattungszeitraums und, wenn Sie weitere Jahre gekauft haben, ab 14 Tagen vor dem Ende der Updates.

Studio versucht es einmal beim Start, höchstens einmal am Tag. Ein fehlgeschlagener Versuch unterbricht Sie nicht. Gesendet wird nur, was die Option nennt.

Wird ein Abonnement nicht verlängert, funktioniert alles noch 14 Tage weiter. Die Lizenz zeigt **In Kulanzfrist** und die verbleibenden Tage. Danach ist Studio die Edition Learning.

Wenn eine Lizenz endet

Ablauf, Kündigung oder eine nicht bestandene Prüfung führen immer nur zurück zur Edition Learning:

- Ihre Projekte öffnen sich wie zuvor;
- bereits erzeugter G-Code lässt sich weiter exportieren und an die Maschine senden;
- die bezahlten Funktionen warten, bis Sie verlängern.

Studio löscht nie Ihre Dateien und sperrt nie Ihre Daten.

Updates einer Kauflizenz

Eine Kauflizenz gilt für die Versionen, die innerhalb von 12 Monaten nach dem Kauf oder nach einer bezahlten Verlängerung der Updates erscheinen. Eine spätere Version läuft als Edition Learning, und die Lizenz zeigt **Updates abgelaufen**. Nutzen Sie weiter die letzte Version, für die sie gilt, oder kaufen Sie auf der Lizenzseite ein weiteres Jahr Updates. Siehe Studio aktualisieren.

Eine Lizenz auf einen anderen Computer übertragen

1. Geben Sie auf der Lizenzseite den Aktivierungscode ein und wählen Sie „Meine Computer anzeigen“.
2. Wählen Sie „Diesen Computer freigeben“ neben dem Computer, den Sie nicht mehr nutzen.
3. Aktivieren Sie die Lizenz auf dem neuen Computer.

Sie können innerhalb von 30 Tagen bis zu drei Computer freigeben. Für mehr schreiben Sie an hello@twinwright.com.

Neue Hardware oder ein neu installiertes System ändert den Fingerabdruck. Studio meldet dann **Diese Lizenz ist an einen anderen Computer gebunden.** Geben Sie den alten Eintrag frei und aktivieren Sie erneut. Bis dahin läuft Studio als Edition Learning.

Aktivierungscode verloren

Nutzen Sie **Aktivierungscode verloren?** auf der Lizenzseite. Fragt die Seite nach einer E-Mail-Adresse, geben Sie die ein, mit der Sie gekauft haben: Jede damit gekaufte Lizenz erhält per E-Mail einen neuen Code. Bittet die Seite Sie, uns zu schreiben, schreiben Sie an hello@twinwright.com von der Adresse aus, an die Ihr Aktivierungscode geschickt wurde; wir prüfen das und senden Ihnen einen neuen Code. In beiden Fällen aktiviert der alte Code nicht mehr, bereits aktivierte Computer arbeiten weiter.

Die Uhr des Computers

Wird die Uhr um mehr als zwei Tage zurückgestellt, meldet Studio **Die Uhr dieses Computers zeigt eine frühere Zeit als beim letzten Mal.** Es zählt die spätere Zeit, keine Lizenz und keine Testphase erhält dadurch Tage zurück. Stellen Sie Datum und Uhrzeit richtig ein.

Studio speichert die zuletzt gesehene Zeit an zwei Stellen. Es zählt die spätere der beiden, daher bringt das Löschen einer davon keine Tage zurück.

Zeigt eine Lizenz oder eine Testphase ein zu frühes Ende, nachdem die Uhr einmal auf ein Datum in der Zukunft gestellt war, stellen Sie Datum und Uhrzeit richtig ein und schreiben Sie an hello@twinwright.com.

Für Administratoren: die Befehlszeile

Mit diesen Optionen öffnet das Programm kein Fenster. Der Ordner ist optional; ohne ihn gilt der Lizenzordner des Computers.

```
TwinwrightStudio.exe --licence-status [FOLDER] [--json]
TwinwrightStudio.exe --licence-request [FOLDER]
TwinwrightStudio.exe --licence-import FILE [FOLDER]
```

- `--licence-status` listet jede Lizenzdatei und Testphase auf, ob sie gilt und warum; `--json` gibt alle Felder aus.
- `--licence-request` gibt den Anforderungscode dieses Computers aus.
- `--licence-import` prüft eine Lizenzdatei und kopiert sie in den Lizenzordner.

Unter Linux heißt das Programm `twinwright-studio`. KI-Clients können denselben Status mit dem schreibgeschützten MCP-Werkzeug `licence_status` lesen.

Was Studio sendet und was unser Server speichert: Datenschutz und was Studio sendet.

Ihre Daten und Backups

Studio legt Ihre Arbeit in Ihren Dokumenten ab und den Zustand der Maschinen in einem eigenen Ordner. Dieses Kapitel listet auf, was wo liegt, was Sie sichern sollten und wie Sie auf einen anderen Computer umziehen.

Was wo liegt

Zwei Ordner enthalten alles, was Ihnen gehört:

- `Dokumente\Twinwright Studio`: Ihre Arbeit;
- `%LOCALAPPDATA%\Twinwright\Studio`: der eigene Zustand des Programms.

Aufträge, Programme und ihre Versionen

`Dokumente\Twinwright Studio\programs\<program>\` enthält für jede Version eines Programms eine Datei (`v0001.nc`) und einen Vermerk, wann, von wem und warum sie gespeichert wurde (`v0001.json`). Der CAM-Auftrag, der ein Programm erzeugt hat, wird daneben versioniert (`job-v0001.json`). Studio dünnt sie nie aus.

Mit **Extras > Einstellungen… > Dateien** lässt sich der Ordner mit den Programmen an einen anderen Ort legen (ab dem nächsten Start).

Berichte

`Dokumente\Twinwright Studio\Reports\`: Prüfberichte (PDF und CSV), Einrichteblätter (HTML) und Diagnosepakete (ZIP).

Die Maschinen und ihre Welten

Jede Maschine hat im Zustandsordner eine Welt: den Tisch, das Werkstück mit seinen Schnitten, die Werkzeuge und die Werkzeugtabelle, die Nullpunktverschiebungen, die Einstellungen der Steuerung, die Temperaturen, die Betriebsstunden und die Kalibrierung des Messtasters. Studio schreibt sie während der Arbeit, jedes Mal wenn die Maschine nach einer Änderung zwei Sekunden stillsteht, und noch einmal, wenn Sie das Fenster schließen. Die Welt der ersten Maschine liegt in `world\`, die der anderen in `machines\<id>\world\`.

Einstellungen

`settings.json` im Zustandsordner enthält alles aus **Extras > Einstellungen…**; `window.json` die Position und Größe des Fensters.

Das Prüfprotokoll

`Dokumente\Twinwright Studio\logs\` enthält eine Datei pro Tag (`audit-YYYY-MM-DD.jsonl`). Die Einträge sind verkettet, sodass eine Änderung oder eine Lücke auffällt, wenn Sie die Integrität des Prüfprotokolls überprüfen (**Extras > Einstellungen… > Protokoll > Integrität überprüfen**). Standardmäßig behält das Prüfprotokoll 180 Tage und höchstens 2 GB; **Extras > Einstellungen… > Protokoll** ändert das. Kopieren Sie den ganzen Ordner, damit die Kette lückenlos bleibt.

Was Sie sichern sollten

Schließen Sie Studio und kopieren Sie dann beide Ordner: `Dokumente\Twinwright Studio` und `%LOCALAPPDATA%\Twinwright\Studio`.

Zwei Dinge wandern bei einer Kopie nicht mit:

- die Schlüssel der KI-Dienste: Sie sind für Ihr Windows-Konto auf diesem Computer verschlüsselt. Geben Sie sie auf dem neuen Computer erneut ein;
- die Lizenz: Sie gehört dem Computer. Aktivieren Sie sie auf dem neuen erneut (siehe Lizenzierung).

Auf einen anderen Computer umziehen

1. Schließen Sie auf dem alten Computer Studio und kopieren Sie beide Ordner.
2. Installieren Sie Studio auf dem neuen Computer, starten Sie es einmal und schließen Sie es dann wieder.
3. Kopieren Sie beide Ordner an dieselben Stellen auf dem neuen Computer.
4. Falls Sie in den Einstellungen den Ordner mit den Programmen oder den Ordner des Prüfprotokolls verlegt haben, prüfen Sie, ob diese Ordner vorhanden sind.
5. Starten Sie Studio, aktivieren Sie die Lizenz und geben Sie die KI-Schlüssel erneut ein.
6. Verbinden Sie Ihre KI-Clients erneut (**KI > KI-Clients verbinden…**).

Eine Maschine neu aufsetzen

Das Zurücksetzen auf die Werkseinstellungen versetzt die Maschine in Ihrem Fenster in den Auslieferungszustand zurück: ihre Welt wird gelöscht (das Werkstück, die Werkzeuge, die Nullpunktverschiebungen, die Einstellungen der Steuerung, die Kalibrierung des Messtasters) und Studio startet neu.

1. Wählen Sie **Datei > Werkseinstellungen…** (ab der Stufe Techniker).
2. Wählen Sie den Eintrag innerhalb von 5 Sekunden erneut, um zu bestätigen.

Zurückgesetzt wird nur diese Maschine. Ihre Programme, Berichte, das Prüfprotokoll, die Einstellungen und die anderen Maschinen bleiben erhalten. Das Zurücksetzen wird im Prüfprotokoll aufgezeichnet.

Studio aktualisieren

Studio sucht nicht selbst nach neuen Versionen und aktualisiert sich nie selbst. Sie entscheiden, wann Sie aktualisieren.

Herausfinden, ob es eine neue Version gibt

1. Wählen Sie in Studio **Hilfe > Info**. Ergebnis: Die Zeile unter dem Programmnamen zeigt Ihre Version, zum Beispiel „Version 1.0.0 · erstellt am 2026-10-10 · Commit 4f1c2ab“.
2. Öffnen Sie die Download-Seite twinwright.com/de/download. Ergebnis: Die Seite nennt die aktuelle Version und die SHA-256-Prüfsumme des Installationsprogramms.

Unterscheiden sich die Versionen, steht auf der Download-Seite die neuere.

Ein Update installieren

Sie brauchen das neue Installationsprogramm. Administratorrechte brauchen Sie nicht: Studio wird für Ihr Windows-Konto installiert.

1. Laden Sie das Installationsprogramm von der Download-Seite herunter.
2. Optional: Vergleichen Sie seine SHA-256-Prüfsumme mit der auf der Seite. Führen Sie dazu in PowerShell `Get-FileHash` mit dem Dateinamen aus.
3. Starten Sie das Installationsprogramm. Meldet Windows einen unbekannten Herausgeber, führen Sie es nicht aus, sondern schreiben Sie an hello@twinwright.com.
4. Ist Studio geöffnet, schließen Sie es, wenn das Installationsprogramm darum bittet.
5. Folgen Sie dem Installationsprogramm. Ergebnis: Die neue Version ersetzt das Programm im selben Ordner.

Ihre Einstellungen, KI-Schlüssel, Programmversionen, das Prüfprotokoll und die Lizenz bleiben, wie sie sind: Sie liegen nicht im Programmordner. Verbundene KI-Clients funktionieren weiter, weil sich der Pfad des Programms nicht ändert.

Für viele Computer läuft das Installationsprogramm auch ohne Rückfragen: `setup.exe /S`. Siehe Studio installieren.

Für welche Versionen Ihre Lizenz gilt

Lizenz	Versionen mit ihren kostenpflichtigen Funktionen
Edition Learning	Alle Versionen
Abonnement	Alle Versionen, solange das Abonnement läuft
Kauflizenz, MC-1-Besitzer	Versionen, die innerhalb von 12 Monaten nach dem Kauf oder nach einer bezahlten Verlängerung der Updates erscheinen

Eine Version, die nach dem Ende der Updates einer Lizenz erscheint, läuft als Edition Learning. Die Lizenz zeigt dann **Updates abgelaufen**. Prüfen Sie das Datum vor dem Update unter **Hilfe > Lizenz…**. Bewahren Sie das Installationsprogramm der letzten Version auf, für die Ihre Lizenz gilt.

Um weiter zu aktualisieren, kaufen Sie auf der Lizenzseite ein weiteres Jahr Updates. Siehe Lizenzen, Arbeitsplätze und Aktivierung.

Computer ohne Internet

1. Laden Sie das Installationsprogramm auf einem Computer mit Internet herunter.
2. Bringen Sie es zum Computer ohne Internet, zum Beispiel auf einem USB-Stick.
3. Installieren Sie es dort wie oben.

Eine neuere Version ändert die Lizenzdatei nicht. Verlängern Sie wie bisher: Einen Computer ohne Internet aktivieren.

Zu einer früheren Version zurückkehren

1. Sichern Sie Ihre Dateien vor dem Update.
2. Deinstallieren Sie Studio: in den Windows-Einstellungen unter „Apps“.
3. Installieren Sie das Installationsprogramm der früheren Version.

> **Hinweis**
>
> Eine frühere Version öffnet unter Umständen keine Projekte oder Einstellungen, die eine spätere gespeichert hat. Behalten Sie die Sicherung, bis Sie sicher sind.

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

Datenschutz und was Studio sendet

Studio arbeitet auf Ihrem Computer. Studio selbst sendet nur, was das Starten einer Testphase und das Aktivieren und Verlängern einer kostenpflichtigen Lizenz braucht. Es gibt keine Telemetrie, keine Suche nach Updates und kein automatisches Hochladen von Abstürzen. Wenn Sie den KI-Assistenten mit einem Cloud-Dienst verbinden, gehen Ihre Nachrichten und was Sie anhängen an diesen Dienst: siehe Der KI-Assistent und Cloud-Dienste. Dieses Kapitel zeigt, was hinausgeht, was auf dem Computer bleibt und wie Sie es entfernen. Verbindlich ist die Datenschutzerklärung der Website, twinwright.com/de/privacy.

Was Studio sendet

Wann	An wen	Was
Sie starten die Testphase einer kostenpflichtigen Edition	Den Lizenzserver von Twinwright, über HTTPS	Die Edition, einen Hash des Fingerabdrucks des Computers, die Studio-Version
Sie aktivieren eine kostenpflichtige Lizenz	Den Lizenzserver von Twinwright, über HTTPS	Den Aktivierungscode, einen Hash des Fingerabdrucks des Computers, die Studio-Version
Studio verlängert eine kostenpflichtige Lizenz (siehe Verlängerung)	Den Lizenzserver von Twinwright, über HTTPS	Die Lizenznummer, denselben Hash, die Studio-Version

Mehr sendet Studio selbst nicht an Twinwright. Der Fingerabdruck verlässt den Computer nie im Klartext, nur sein Hash. Ein offline aktivierter Computer sendet überhaupt nichts.

Drei Dinge gehen nur hinaus, wenn Sie sie selbst einrichten:

- **Der KI-Assistent, mit einem Cloud-Dienst verbunden.** Er sendet Ihre Nachrichten und was Sie anhängen an diesen Dienst; siehe unten.
- **Ein KI-Client, den Sie über MCP verbinden**, etwa Claude oder ChatGPT. Der Client sendet, was er aus dem Zwilling liest, an seinen eigenen Dienst, nach dessen Bedingungen.
- **Eine Webseite, die Studio öffnet**, etwa die Lizenzseite. Ihr Browser öffnet sie; Studio fügt nichts hinzu.

Der KI-Assistent und Cloud-Dienste

Der KI-Assistent arbeitet mit dem Anbieter, den Sie in ihm wählen: einem Cloud-Dienst wie Anthropic, OpenAI oder DeepSeek, mit Ihrem eigenen Konto und API-Schlüssel, oder einem lokalen Modell.

Wenn Sie einem Cloud-Dienst eine Nachricht senden, sendet Studio von Ihrem Computer und mit Ihrem Schlüssel an diesen Dienst:

- Ihre Nachricht, mit der bisherigen Unterhaltung;
- Maschine, Aufspannung, Programm (mit Zeilennummern) und die von den Prüfungen gefundenen Probleme aus dem Fenster, solange **Programm, Aufspannung und Prüfergebnisse des Fensters anhängen (wenn geändert)** angehakt ist. Das Kästchen ist anfangs angehakt. Studio hängt diese Angaben an die erste Nachricht an und danach nur, wenn sie sich geändert haben;
- die Ergebnisse der Tools, die der Assistent in Studio aufruft, etwa die Prüfung oder Simulation eines Programms, den Zustand der Maschine oder Einträge des Protokolls, und Bilder von Ergebnissen an Dienste, die Bilder annehmen.

Bevor die erste Nachricht an einen Cloud-Dienst geht, zeigt Studio einmal, was es senden wird.

Der Dienst behandelt all das nach seinen eigenen Bedingungen; Twinwright erhält nichts davon. Um weniger zu senden, entfernen Sie vor dem Senden den Haken, oder beginnen Sie mit **Neue Unterhaltung** von vorn.

Mit einem lokalen Modell (LM Studio, Ollama) bleibt all das auf dem Computer.

Auf diesem Computer hält das Prüfprotokoll die ersten 180 Bytes jeder Nachricht fest (siehe Das Prüfprotokoll), und die API-Schlüssel werden verschlüsselt gespeichert (siehe Was Studio auf dem Computer speichert).

Was Studio auf dem Computer speichert

Was	Wo (Windows)	Wer es sieht
Einstellungen, der Zustand des Zwillings, KI-Schlüssel	`%LOCALAPPDATA%\Twinwright\Studio`	Ihr Windows-Konto
Programmversionen	`Dokumente\Twinwright Studio`	Ihr Windows-Konto
Das Prüfprotokoll	`Dokumente\Twinwright Studio\logs`, sofern die Einstellungen keinen anderen Ordner nennen	Ihr Windows-Konto
Prüfberichte, Einrichteblätter, Diagnosedateien	`Dokumente\Twinwright Studio\Reports`	Ihr Windows-Konto
Absturzberichte	`%LOCALAPPDATA%\Twinwright\Studio\crash`, nach 30 Tagen gelöscht	Ihr Windows-Konto
Lizenzdateien, Testphasen eingeschlossen	`%ProgramData%\Twinwright\licence`	Alle auf dem Computer

KI-Schlüssel werden für Ihr Windows-Konto verschlüsselt gespeichert (Windows DPAPI). Andere Konten auf demselben Computer können sie nicht lesen.

Das Prüfprotokoll

Das Prüfprotokoll ist eine Datei pro Tag, `audit-YYYY-MM-DD.jsonl`, im Protokollordner. Es hält fest:

- was die KI und Sie die Maschine haben tun lassen;

- die ersten 180 Bytes jeder Nachricht im KI-Bereich;
- Lizenzprüfungen: Lizenznummer, Edition und Ergebnis, ohne den Namen des Lizenznehmers.

Schlüssel, Token und Passwörter werden als `***` geschrieben. In den Einstellungen legen Sie fest, wie viel das Protokoll festhält und wie lange es aufbewahrt wird. Das Protokoll bleibt auf dem Computer; Studio lädt es nie hoch.

Wenn Studio unerwartet beendet wird

Im Moment des Fehlers schreibt Studio einen kleinen Bericht: wo der Fehler auftrat, die Folge der Aufrufe bis dahin und die Liste der Programmmodule. Speicherinhalte schreibt es nicht.

Beim nächsten Start ergänzt Studio die Versionsinformationen und die letzten 200 Zeilen seines Protokolls, bereinigt:

- Pfade und Dateinamen werden durch Platzhalter ersetzt;
- Namen und E-Mail-Adressen werden entfernt;
- kein G-Code, kein KI-Gespräch, kein Lizenzcode und kein Fingerabdruck.

Studio fragt dann einmal. Sie können den Ordner des Berichts öffnen, den Bericht als Datei speichern oder die Meldung schließen. Es wird nichts gesendet. Wenn Sie uns bei der Ursachensuche helfen möchten, senden Sie die Datei selbst an hello@twinwright.com.

Diagnoseinformationen

Wenn Sie uns um Hilfe bitten, fragen wir eventuell nach Diagnoseinformationen. Wählen Sie im Menü **Hilfe** „Diagnoseinformationen sammeln“. Studio speichert eine ZIP-Datei mit den angekreuzten Punkten im Berichtsordner:

Punkt	Anfangs angekreuzt
Version, System, Grafik, Sprache und Einheiten	Ja
Einstellungen, ohne die KI-Schlüssel	Ja
Das Maschinenprofil und die `$`-Einstellungen der Steuerung	Ja
Die letzten 7 Tage des Protokolls und die Absturzberichte, bereinigt wie oben	Ja
Der Lizenzstatus: Edition und Enddatum, ohne Codes und Fingerabdruck	Ja
Das aktuelle Programm und die Aufspannung: die Daten Ihres Teils	Nein

Öffnen und lesen Sie die Datei, bevor Sie sie senden. Es wird nichts automatisch gesendet.

Was im Netzwerk lauscht

Die Dienste von Studio für KI-Clients (MCP) und für Sender-Programme lauschen nur auf diesem Computer (`127.0.0.1`). Andere Computer erreichen sie nicht.

Die Befehlszeilenoption `--bind any` öffnet die Netzwerkports des Zwillings, etwa den für Sender-Programme, für Ihr Netzwerk. Der MCP-Dienst bleibt lokal.

> ⚠ **Warnung**
>
> Im Schattenmodus leitet der Port für Sender-Programme Befehle an die reale Maschine weiter. Mit `--bind any` könnte jeder in Ihrem Netzwerk, der diesen Port erreicht, die Maschine bewegen. Verwenden Sie `--bind any` nur für den Zwilling und nur in einem Netzwerk, das Sie kontrollieren.

Was unser Server speichert

Zu jeder kostenpflichtigen Lizenz speichert der Lizenzserver:

- die Hashes von bis zu drei Computern je Arbeitsplatz, jeweils mit dem Datum der ersten und der letzten Meldung, damit Sie auf der Lizenzseite einen freigeben können;
- den Aktivierungscode nur als Hash: Nicht einmal wir können ihn zurücklesen;
- 90 Tage lang ein Protokoll der Testphasen, Aktivierungen, Verlängerungen und Änderungen, mit dem Hash des Computers, der Studio-Version und Ihrer auf den Netzanteil gekürzten IP-Adresse.

Zu jeder Testphase speichert er den Hash des Computers mit der Edition und den Daten der Testphase, damit jeder Computer jede Testphase einmal hat.

Nichts über Ihre Arbeit erreicht den Server.

Entfernen, was Studio speichert

1. Geben Sie den Computer auf der Lizenzseite frei, damit er nicht mehr zählt (siehe Eine Lizenz auf einen anderen Computer übertragen).
2. Deinstallieren Sie Studio.
3. Löschen Sie die Ordner aus der Tabelle oben, die Sie nicht mehr brauchen. Ergebnis: Von Studio bleibt nichts auf dem Computer.

Die Deinstallation entfernt das Programm, nicht Ihre Dateien: Ihre Projekte, Programmversionen, Protokolle und Berichte bleiben, bis Sie sie löschen.

Labore, Unterrichtsräume und gemeinsam genutzte Computer

Dieses Kapitel richtet sich an die Person, die Studio auf Computern einrichtet, die mehrere Personen nutzen: im Labor, in der Werkstatt oder im Unterrichtsraum.

Eine Lizenz für den ganzen Computer

Der Lizenzordner gehört zum Computer, nicht zu einem Windows-Konto (siehe Der Lizenzordner). Eine auf einem Computer aktivierte Lizenz gilt für alle, die sich daran anmelden.

Ein kostenpflichtiger Arbeitsplatz wird jeweils von einer Person genutzt, auf bis zu drei Computern, und ein Computer, den mehrere Personen gemeinsam nutzen, zählt einmal. Daher:

- Geben Sie einem gemeinsam genutzten Computer, etwa dem neben der Maschine, einen eigenen Arbeitsplatz.
- Wer die kostenpflichtigen Funktionen auch auf eigenen Computern nutzt, braucht für diese weitere Arbeitsplätze.

Aktivieren Sie den gemeinsam genutzten Computer einmal, mit dem Konto der Person, die ihn betreut.

Was jede Person für sich hat

Studio wird für jedes Windows-Konto installiert. Jede Person hat ihre eigenen:

- Einstellungen und KI-Schlüssel: Die Schlüssel sind für dieses Konto verschlüsselt, andere Konten können sie nicht lesen;
- Programmversionen, Berichte und das Prüfprotokoll, im eigenen Ordner `Dokumente\Twinwright Studio`.

Die Testphase einer Edition zählt einmal für den Computer, für alle darauf: Der Lizenzserver gibt jedem Computer von jeder Edition eine Testphase.

Zur Installation auf vielen Konten und Computern siehe Studio installieren.

Unterrichtsräume

Die Edition Learning ist auf jedem Computer kostenlos, zum Lernen und Unterrichten. Es muss nichts aktiviert werden: Installieren Sie Studio, und der ganze Zwilling, der Editor und das CAM für alltägliche Teile funktionieren.

Geben Sie, wenn möglich, jeder Schülerin und jedem Schüler ein eigenes Windows-Konto. Dann bleiben Einstellungen, Programme und Protokolle getrennt.

Für den Unterricht mit den kostenpflichtigen Funktionen erhalten Schulen und Bildungseinrichtungen Pro-Arbeitsplätze. Schreiben Sie an hello@twinwright.com.

Computer, die bei jedem Start zurückgesetzt werden

Manche Laborcomputer kehren bei jedem Start zu einem gespeicherten Abbild zurück, wodurch auch der Lizenzordner geleert wird. Bei einem solchen Computer:

1. Aktivieren Sie den Computer einmal offline (siehe Einen Computer ohne Internet aktivieren) und bewahren Sie seine Lizenzdatei auf.
2. Legen Sie die Datei in den Lizenzordner des Abbilds, oder importieren Sie sie bei jedem Start:

```
TwinwrightStudio.exe --licence-import D:\licences\lab-pc-3.twlic
```

Eine Lizenzdatei gilt für einen Computer. Jeder Computer des Raums braucht seinen eigenen Anforderungscode und seine eigene Datei.

Computer ohne Internet

Aktivieren Sie sie offline und verlängern Sie von Hand: vor dem Ende eines Abonnementzeitraums und einmal nach den ersten 30 Tagen einer Kauflizenz. Siehe Einen Computer ohne Internet aktivieren.

Wenn in den Lizenzordner nicht geschrieben werden kann

Unter Windows kann jedes Konto `%ProgramData%\Twinwright\licence` anlegen. Das Konto, das ihn anlegt, kann seine Dateien ändern; die anderen Konten können sie lesen. Das genügt: Eine Person importiert die Lizenz, und alle nutzen sie.

Wenn die Richtlinien Ihrer Organisation das nicht zulassen:

- legt ein Administrator den Ordner an und gibt den Benutzern Schreibrechte; oder
- die Systemumgebungsvariable `TWINWRIGHT_LICENCE_DIR` nennt einen Ordner, den alle lesen können.

Unter Linux ist der Ordner `/etc/twinwright/licence`, wenn Ihr Konto ihn anlegen oder darin schreiben kann; sonst verwendet Studio Ihren eigenen `~/.local/share/Twinwright/licence` (unter `$XDG_DATA_HOME`, wenn gesetzt), nur für Sie. Um eine Lizenz unter den Benutzern des Computers zu teilen, legen Sie den Ordner einmal an mit `sudo install -d -m 1777 /etc/twinwright/licence`.

KI auf einem gemeinsam genutzten Computer

Jede Person verbindet ihren eigenen KI-Anbieter oder KI-Client, mit ihrem eigenen Schlüssel. Das Prüfprotokoll jedes Kontos hält fest, was diese Person und die KI die Maschine haben tun lassen. Siehe Datenschutz und was Studio sendet.

Technical manual

Die Modelle des Zwillings und ihre Grenzen

Der Zwilling ist die MC-1 als Programm: Twinwrights eigener Kernel läuft darin, und physikalische Modelle stehen für die Teile der Maschine rund um die Steuerung. Jede Zahl, die der Zwilling über die physische Welt angibt, ist eine Schätzung aus einem physikalischen Modell. Seine Parameter stammen aus der Konstruktion der Maschine, aus den Datenblättern der Teile und aus veröffentlichten Werten; jede MC-1 soll bei der Inbetriebnahme vermessen und ihr Zwilling auf die Messwerte eingestellt werden (geplant). Jeder Abschnitt unten endet mit dem, worauf seine Parameter beruhen.

Entwurf (Runde 14): Der Text wird noch geschrieben; die Abschnitte können noch wachsen.

Die Steuerung

Die Steuerung im Zwilling ist Twinwrights eigener Kernel, aus demselben Quelltext gebaut wie die Firmware der Maschine. Modelliert wird die Platine um ihn herum: ihre Timer und Schrittimpulse, ihre Eingänge mit deren Entprellung, ihre Verbindung zum Computer (der serielle USB-Port und sein Handshake) und die Zeit, die der Vordergrund der Steuerung braucht.

Grundlage: der eigene Code des Kernels; die Zeit des Vordergrunds ist eine Schätzung, bis sie auf der Platine gemessen wird.

Bewegung und Antriebe

Jede Achse wird von einem Schrittantrieb mit geschlossenem Regelkreis und einem Kugelgewindetrieb angetrieben. Der Zwilling modelliert sie in vier Detailstufen; Standard ist ein Dreimassenmodell:

- die Lageregelung des Antriebs: die magnetische Feder des Motors um die Sollposition, sein mit der Drehzahl abfallendes Drehmoment, das Außertrittfallen bei zu hoher Last, der Alarm des Antriebs, wenn der Schleppfehler zu groß wird;
- die Reibung in den Führungen, in der Mutter und in den Lagern sowie die Schwerkraft in Z;
- die Nachgiebigkeit der Übertragung zwischen Motor, Gewindespindel und Tisch, das Umkehrspiel der Mutter, der Rahmen unter der X-Achse;
- die Leistungsversorgung der Antriebe: die Not-Halt-Kette, das Schütz, die Zeit, die die Antriebe zum Hochfahren brauchen.

Grundlage: die Konstruktion (Massen, Trägheiten, Steifigkeit aus Handrechnung und Finite-Elemente-Analyse), die Datenblätter der Antriebe und Motoren; Reibung und die internen Parameter des Antriebs sind Annahmen, bis sie gemessen werden.

Geometrie- und Genauigkeitsmodell

Wenn es eingeschaltet ist, verschiebt der Zwilling die Werkzeugspitze um die geometrischen Fehler der Maschine: den Steigungsfehler jeder Achse und seinen periodischen Anteil, zwei Geradheitsfehler und drei Winkelfehler, die über ihre Abbe-Arme wirken, die Rechtwinkligkeit zwischen den Achsen, das Umkehrspiel der Mutter sowie die Wärmedehnung der Gewindespindeln und der Spindel. Die Endschalter sehen die Schlitten, der Werkzeuglängensensor und der Messtaster sehen die Werkzeugspitze, wie an der Maschine. Das Modell ist standardmäßig aus: Der nominelle Zwilling ist ideal.

Grundlage: Tabellen einer typischen aufgebauten Maschine, aus den Genauigkeitsklassen der Teile abgeleitet. Sie sind nicht die Spezifikation der Maschine; sie werden bei der Inbetriebnahme durch die Messung der Maschine nach ISO 230-2 ersetzt (geplant).

Wärme

Die Spindel und die Kugelgewindetriebe erwärmen sich im Betrieb und kühlen in Ruhe ab; der Zwilling behält ihre Temperaturen zwischen den Sitzungen bei und verschiebt je nach Temperatur die Werkzeugspitze und die Steigung der Gewindespindeln.

Grundlage: thermische Modelle erster Ordnung mit angenommenen Parametern, bis das Warmlaufen der Maschine gemessen ist.

Spindel und Antrieb (Frequenzumrichter)

Der Spindelantrieb ist so modelliert wie der der Maschine: Die Steuerung spricht über RS-485 mit ihm, im eigenen Protokoll des Antriebs, Byte für Byte mit der eingestellten Geschwindigkeit, und der Antrieb antwortet so, wie sein Handbuch es beschreibt, einschließlich seiner Parameter, die bestimmen, ob er zuhört. Dahinter: das Drehmoment des Motors aus seinem Schlupf, sein Strom und die Überstrom-, Überlast- und Unterspannungsabschaltungen des Antriebs, der Zwischenkreis, der den Antrieb über einen kurzen Spannungseinbruch hinweg am Leben hält, die Wasserpumpe.

Grundlage: die Daten des Antriebs und des Motors; die Kapazität des Zwischenkreises, der magnetisierende Anteil des Stroms und die Reibung sind Annahmen, bis sie gemessen werden.

Der Schnitt

Das Rohteil ist ein Höhenfeld mit feineren Bereichen dort, wo Sie sie anfordern. Jede Bewegung des Werkzeugs trägt ab, was die Form des Werkzeugs überstreicht, sodass die Form des Werkstücks aus dem Programm selbst entsteht.

Die Schnittkraft ergibt sich aus den Schnittkoeffizienten des Materials, die sich mit der Spanungsdicke und der Schnittgeschwindigkeit ändern und Zahn für Zahn an der tatsächlichen Form des Werkzeugs berechnet werden (Flachfräser, Kugelfräser, Torusfräser, V-Fräser). Aus der Kraft schätzt der Zwilling die Last und die Leistung der Spindel, die Durchbiegung des Werkzeugs und das Risiko eines Werkzeugbruchs: die Biegespannung am Spannutgrund gegenüber der Festigkeit des Hartmetalls.

Grundlage: veröffentlichte Schnittdaten für die Materialien (die Quelle jedes Materials steht im Maschinenprofil), ein scharfes Werkzeug, typische Werkzeuggeometrie.

Rattern

Für jeden Schnitt berechnet der Zwilling aus den Schwingungsmoden des Werkzeugs und der Maschine und der Geometrie des Schnitts die Tiefe, bei der der Schnitt bei der verwendeten Spindeldrehzahl zu rattern beginnen würde, und wählt eine Spindeldrehzahl, bei der der Schnitt stabil ist. Außerdem schätzt er aus der erzwungenen Schwingung des Schnitts, wo die Wände am Ende liegen.

Die Werkzeugspitze hat in jeder Richtung drei Moden: die Struktur der Maschine, den Werkzeughalter in Spannzange und Spindel und die Biegung des Werkzeugs selbst. Die Frequenz der Biegemode ergibt sich aus der Form des Werkzeugs – seinen Schneiden, seinem Hals und Schaft und seiner Auskraglänge: etwa 14 kHz für einen Schaftfräser von 6 mm mit 20 mm Auskraglänge, über 100 kHz für einen von 0,2 mm. Werkzeuge unter etwa 1 mm rattern, wenn überhaupt, mit ihrer eigenen Biegung, bei einer Tiefe, die kaum von der Spindeldrehzahl abhängt; der Zwilling berechnet diese Tiefe für jeden Schnitt im Zeitbereich.

Grundlage: die Moden des Rahmens aus der Finite-Elemente-Analyse, eine angenommene Mode des Halters und die Biegung des Werkzeugs, aus seiner Geometrie mit der Balkentheorie geschätzt; die Dämpfung aller drei ist angenommen (3 %). Noch keine davon ist gemessen: Sie sollen durch einen Klopftest an der aufgebauten Maschine gemessen werden, und bis dahin sind die stabilen Tiefen Schätzungen. Die Dämpfung des Schnitts selbst ist nicht modelliert: Für kleine Werkzeuge liegt der Ratterkennwert auf der sicheren Seite.

Rundlauffehler

Ein Fräser, dessen Achse gegenüber der Spindelachse versetzt ist, schneidet mit seinem äußersten Zahn: Bohrungen werden größer und Zapfen kleiner (Befund 187). Der Zwilling schneidet mit dieser Hüllkontur; CAM berücksichtigt sie bei Passungen. Ein Werkzeug kann seinen eigenen Rundlauffehler in seiner Angabe mitbringen – die Klasse einer Präzisionsspannzange oder einen gemessenen Wert (`tir3`: sein Rundlauffehler in Mikrometern); sonst gilt der der Maschine.

Grundlage: ein angenommener Rundlauffehler, bis er an der Maschine gemessen wird.

Werkzeugverschleiß

In 1.0 ist jedes Werkzeug ein neues: Die Form, die es schneidet, und seine Kräfte sind die eines scharfen Werkzeugs, wie lange es auch schon geschnitten hat. Für eine spätere Version ist geplant, für jedes Werkzeug mitzuzählen, wie viel Standzeit ein Programm verbraucht. Werkzeuge, die Aluminium zerspanen, werden meist zuerst gewechselt, weil Aluminium an der Schneide haftet und das Werkstück Grate oder eine schlechtere Oberfläche zeigt, nicht weil die Freifläche verschlissen ist.

Der Messtaster

Der Messtaster ist ein Werkzeug, dessen Kugel den Messtastereingang schaltet, wenn sie das Werkstück berührt. Der Zwilling berechnet die Berührung exakt aus der Form des Rohteils und aus den geschnittenen Flächen, Schrägen eingeschlossen. Zwei Modelle: ein idealer Messtaster, der schaltet, sobald die Kugel berührt, und ein typischer Messtaster mit richtungsabhängigem Vorlauf, drei Keulen, Streuung und einem Taststift, der bricht, wenn er zu weit gedrückt wird.

Grundlage: typische Werte für einen schaltenden Messtaster; kalibrieren Sie den Messtaster an einer Ringlehre, bevor Sie damit messen.

Der Z-Freiraum

Für die Aufspannung auf dem Tisch berechnet der Zwilling den höchsten Punkt, der überfahren werden muss, und für das Werkzeug in der Spindel und jedes Werkzeug in der Werkzeugtabelle die Höhe der Werkzeugspitze am oberen Ende von Z. Ein Werkzeug, das die Aufspannung nicht überfahren kann, wird gemeldet, wenn sich die Aufspannung oder das Werkzeug ändert (Befund 188).

Grundlage: die Konstruktion der Maschine und die Werkzeuglängen wie angegeben.

Kollisionen und Spannmittel

Der Zwilling prüft das Werkzeug, seinen Halter und die Spindel gegen das Werkstück, das Spannmittel und die Maschine und warnt vor einer Beinahe-Kollision. Die Spannmittel (Spannpratzen, Schraubstock, Prisma, Klebeband, ein fixiertes Werkstück) halten das Werkstück mit einer Kraft fest; ein Schnitt, der stärker drückt, kann es verrutschen lassen.

Grundlage: die Geometrie der Maschine und der Spannmittel aus der Konstruktion; die Haltekräfte sind Schätzungen.

Was nicht modelliert wird

- die Oberflächenstruktur und die Rauheit über die Lage der Wände hinaus;
- Aufbauschneiden, Späne und ihre Abfuhr;
- dünne Wände, die sich unter dem Schnitt biegen;
- die Eigenwärme des Werkstücks;
- was nach einem Werkzeugbruch geschieht.

Die Abnahmetests, die eine gebaute Maschine brauchen (die Messungen der Antriebe und der Geräusche, die Schnittkräfte, die Oberfläche, die Wärme, die Moden des Rahmens, der Schattenmodus mit einer echten Maschine), stehen in den Entwurfsnotizen des Zwillings (docs/06, Abschnitt 7).

Twinwrights eigener Kernel

Twinwrights eigener Kernel ist die CNC-Steuerungs-Firmware von Twinwright selbst. In Studio steuert er die Maschine des Zwillings. Derselbe Code wird für die Steuerungsplatine der MC-1 gebaut.

Er spricht das serielle Protokoll von Grbl 1.1, sodass G-Code-Sender-Programme mit ihm arbeiten. Einstellungen, Alarme und Fehler behalten die Nummern, die Sender-Programme kennen.

Was er leistet

- **Bewegung:** vorausschauende Planung über 100 Blöcke, Eckengeschwindigkeiten aus `$11` und Kreisbögen auf dem echten Kreis. Die Bewegung läuft in Zeitscheiben von 1 ms. Jede Achse hält ihre maximale Geschwindigkeit (`$110`–`$112`) und Beschleunigung (`$120`–`$122`) ein.
- **Maschinenfunktionen:**
  - Referenzfahrt, Hardware- und Software-Endschalter, Antasten und der Werkzeuglängensensor;
  - der Spindelantrieb (Frequenzumrichter) über Modbus und die Kühlung;
  - Werkzeugwechsel, wie `$341` sie festlegt;
  - die Schutztür mit Parken;
  - Overrides, Tippbetrieb und das Handrad.
- **Programme auf der Steuerung:** Programme in ihrem Dateisystem laufen mit O-Wort-Unterprogrammen und Makros (siehe O-Wörter).
- **G-Code:** RS274/NGC und LinuxCNC. Dazu gehören eine Werkzeugtabelle, Fräserradiuskorrektur, Bahnverschleifung, Ausdrücke und Parameter (siehe G-Code und M-Code).

Das serielle Protokoll, die Echtzeitbefehle und die `$`-Befehle stehen in Protokoll. Jede Einstellung steht in Einstellungen, jeder Alarm in Alarmcodes und jeder Fehler in Fehlercodes.

Im Vergleich mit anderen Grbl-Steuerungen

Sender-Programme, die für Grbl geschrieben sind, arbeiten mit dem Kernel. Dieser Abschnitt listet auf, was Ihnen auffällt, wenn Sie eine andere Grbl-Steuerung gewohnt sind.

Wo der Kernel anhält

In diesen Fällen machen manche Grbl-Steuerungen weiter; der Kernel hält an oder löst einen Alarm aus.

- **Der Spindelantrieb hält von selbst an** (eine Abschaltung, eine Überlast oder seine eigene Stopptaste): Der Vorschub stoppt innerhalb von 0,2 s, und der Kernel löst `ALARM:14` aus.
- **Der Spindelantrieb antwortet nicht:** Die Achsen bremsen zuerst bis zum Stillstand ab, sodass die Position erhalten bleibt. Dann löst der Kernel `ALARM:19` aus.
- **Werkzeugwechsel:**
  - Die Spindel hält bei einem Werkzeugwechsel immer an.
  - Läuft ein Werkzeugwechsel ohne Bestätigung weiter, kommen Spindel und Kühlung wieder so zurück, wie sie waren.
- **Fortsetzen nach Vorschub halt oder der Tür:** Die Spindel erreicht ihre Drehzahl, bevor sich das Werkzeug bewegt.
- **Die Tür öffnet sich wieder, während die Maschine vom Parken zurückkehrt:** Die Maschine hält an und fährt wieder auf die Parkhöhe. Sie wartet auf den nächsten Zyklusstart.
- **Das Sender-Programm geht verloren** (USB abgezogen, der Port geschlossen): Der Auftrag stoppt. `$1030` steuert dies; es ist standardmäßig eingeschaltet.
- **Messtaster- und Referenzschalter:** Störimpulse auf den Leitungen zählen nicht als Berührung.
- **UTF-8-Text in einem Programm** (Chinesisch, „Ø“, „–“) wird als Text gelesen, nie als Echtzeitbefehl.
- **Ein Kreisbogen in R-Form ohne Sehne in seiner Ebene**, zum Beispiel `G91 G2 X0 Z1 R5`, antwortet mit `error:33`.

Mehr G-Code

Manche Grbl-Steuerungen lehnen diese ab oder ignorieren sie; der Kernel führt sie so aus, wie die Normen sie festlegen:

- die Werkzeugtabelle (`G43 Hn`, `G43.2`, `G10 L1/L10/L11`);
- die Fräserradiuskorrektur (`G41`, `G42`);
- Bahnverschleifung (`G64 P`) und Genauhalt (`G61.1`);
- O-Wort-Unterprogramme und Ablaufsteuerung, `#`-Parameter und Ausdrücke;
- `G52`, Polarkoordinaten (`@`, `^`) und `M99`;
- die Zyklen `G85`, `G86` und `G89` zum Bohren und Ausbohren, wobei manche Grbl-Steuerungen sie annehmen und sich nicht bewegen;
- was CAM-Postprozessoren oft schreiben: ein `%`-Satz mit einem Namen, `G49 H0`, Fanuc-Programmnummernsätze (`O1000`) und Centroids `M25`.

Näher am Programm

- **Kreisbögen** folgen dem echten Kreis. Manche Grbl-Steuerungen zerlegen sie in Sehnen, die bis zu 4 × `$12` vom Kreis abweichen.
- **Kreisbögen mit Vorschub im Kehrwert der Zeit** (`G93`) dauern so lange wie programmiert. Manche Grbl-Steuerungen fahren sie um ein Mehrfaches schneller.
- **Verweilzeiten** (`G4`) dauern so lange wie programmiert. Manche Grbl-Steuerungen runden sie auf Schritte von 50 ms.
- **Tieflochbohren mit Entspanen:** Jedes Loch bekommt die Zustellungen, die es braucht. Ist die Tiefe ein ganzzahliges Vielfaches der Zustellung, bohren manche Grbl-Steuerungen von der Rückzugshöhe aus noch eine volle Zustellung.
- **Große Kreisbögen nahe den Software-Endschaltern** lösen keine Fehlalarme aus.

- `M72` stellt die Werkzeuglängenkorrektur wieder her, die `M70` oder `M73` gespeichert hat. Manche Grbl-Steuerungen behalten eine später gesetzte Korrektur, wenn die gespeicherte 0 war und die Korrekturart (`G43.1`) dieselbe ist, sodass dasselbe Programm nach `M72` dort mit einer anderen Werkzeuglängenkorrektur laufen kann.
- **Beschleunigung:** Jede Achse hält `$120`–`$122` ein, auch bei kleinen Kreisbögen. Ein Programm mit sehr vielen kleinen Kreisbögen kann bis zu etwa 2 % länger dauern als auf manchen Grbl-Steuerungen. Ein gewöhnliches Programm dauert etwa 0,1 % weniger.

Protokolldetails

- **Statusberichte:**
  - Bei Vorschub halt oder offener Tür ist der Vorschub in `FS:` 0: Die Achsen stehen.
  - Der Restweg wird gemeldet, bis die Achsen stehen bleiben.
  - Bei einem kritischen Alarm zeigt `Bf:` den Planer als leer an.
  - Ist der Parserzustand in `$10` eingeschaltet, kommt `[GC:]` einmal pro Änderung.
- **Eingabeprüfungen:**
  - Eine fehlerhafte Einstellungsnummer wie `$45x` antwortet mit `error:3`.
  - Eine Ganzzahl-Einstellung, die als `$1=25.0` geschrieben wird, speichert 25; manche Grbl-Steuerungen speichern 250.
  - `M6` und eine Bewegung in einem Satz antworten mit `error:40`.
  - `G65` oder `G66` mit anderen Befehlen in einem Satz antworten mit `error:21`.
- **Alarme und Start:**
  - Nach einem Alarm gibt es keine Pause von 100 ms.
  - Der erste Start mit leerem Einstellungsspeicher antwortet nicht mit `error:7`.
- **Texte und Einstellungen:**
  - Die Beschreibungen der Einstellungen, Alarme und Fehler sind Twinwrights eigene.
  - Der Kernel hat eigene Einstellungen: `$1030` (Stopp, wenn das Sender-Programm verloren geht) und `$1000` und höher für Input Shaping (standardmäßig aus).
- `$39=0` (Legacy-Echtzeitzeichen aus): `?`, `!` und `~` sind Text innerhalb von `$`-Zeilen und Kommentaren und überall sonst Befehle. Der Kernel entscheidet anhand der Bytes, sobald sie eintreffen. Deshalb beantwortet er `?` auch bei offener Tür oder bei Vorschub halt, wo manche Grbl-Steuerungen manchmal nicht antworten.

Nicht unterstützt

- `G65 P7` (Modbus-Nachrichten).
- Fanuc-Systemvariablen. `#4120` und dergleichen sind gewöhnliche Parameter und ergeben beim Lesen 0.
- Gewindeschneiden (`G33`, `G76`, `G33.1`) ohne Spindelencoder. Die MC-1 hat keinen, daher antworten diese mit `error:20`.
- Schruppzyklen für Drehmaschinen (andere Grbl-Steuerungen haben ebenfalls keine).
- Netzwerk, Trinamic-Treiber und andere Fabrikate von Spindelantrieben, die manche Grbl-Steuerungen als Plug-ins anbieten.

Sender-Programme

Die Sender-Programme, die geprüft wurden, und ihre Einstellungen stehen in G-Code-Sender-Programme. Wie Sie eine echte Maschine mit ihrem Zwilling vergleichen, steht in Echte Maschine und Schattenmodus.

G-Code und M-Code

Twinwrights eigener Kernel führt den unten stehenden G-Code aus. Die Codes folgen RS274/NGC und LinuxCNC. Ein Code, den der Kernel nicht unterstützt, wird mit `error:20` beantwortet, und der Satz bewirkt nichts.

Zu den Echtzeitbefehlen (einzelne Bytes wie `?`, `!` und `~`) siehe Protokoll. Wie sich der Kernel von anderen Grbl-Steuerungen unterscheidet, steht in Twinwrights eigener Kernel.

G-Code

Code	Bedeutung	Hinweise
`G0`, `G1`	Eilgang, lineare Vorschubbewegung	
`G2`, `G3`	Kreisbogen im Uhrzeigersinn, gegen den Uhrzeigersinn (I J K oder R), Helix	Das Werkzeug folgt dem exakten Kreis. Ein Kreisbogen in R-Form ohne Sehne in seiner Ebene wird mit `error:33` beantwortet.
`G4 P`	Verweilzeit, P Sekunden	Die Verweilzeit dauert so lange wie programmiert.
`G5`, `G5.1`	Kubischer Spline, quadratischer Spline	Ein `G5` ohne I und J setzt in der Richtung fort, in der die vorige Bewegung endete.
`G7`, `G8`	Durchmessermodus beim Drehen, Radiusmodus	Nur im Drehmodus (`$32=2`).
`G10 L2`, `G10 L20`	Eine Nullpunktverschiebung setzen	
`G10 L1`, `G10 L10`, `G10 L11`	Die Werkzeugtabelle setzen	32 Werkzeuge. Die Tabelle bleibt erhalten, wenn die Steuerung ausgeschaltet ist.
`G17`, `G18`, `G19`	Ebene wählen	
`G20`, `G21`	Einheit Zoll, Einheit Millimeter	
`G28`, `G30`	Eine gespeicherte Position über einen Zwischenpunkt anfahren	
`G28.1`, `G30.1`	Die aktuelle Position speichern	
`G33`, `G33.1`, `G76`	Spindelsynchrone Bewegung, starres Gewindebohren, Gewindeschneidzyklus	Diese brauchen einen Spindelencoder. Die MC-1 hat keinen, daher werden sie mit `error:20` beantwortet.
`G38.2`–`G38.5`	Antasten	Das Ergebnis ist der Punkt, an dem der Messtaster ausgelöst hat, gemeldet als `[PRB:]`. Setzen Sie den Nullpunkt auf Grundlage dieses Punktes, nicht der Stelle, an der die Achsen angehalten haben.
`G40`	Fräserradiuskorrektur aus	
`G41`, `G42`, `G41.1`, `G42.1`	Fräserradiuskorrektur: D aus der Werkzeugtabelle oder `.1` mit einem Durchmesser	Nur in der Ebene G17. Ein- und Ausfahrbewegungen wie in LinuxCNC. Eine Bewegung, die einen Überschnitt verursachen würde, wird abgelehnt.
`G43 Hn`, `G43.2 Hn`	Werkzeuglängenkorrektur aus der Werkzeugtabelle, zusätzliche Korrektur	
`G43.1`	Dynamische Werkzeuglängenkorrektur	
`G49`	Werkzeuglängenkorrektur aus	
`G50`, `G51`	Skalierung aus, ein	
`G52`	Lokales Koordinatensystem	Teilt seine Verschiebung mit `G92`, wie in LinuxCNC.
`G53`	Maschinenkoordinaten für diesen Satz	
`G54`–`G59`, `G59.1`–`G59.3`	Werkstückkoordinatensysteme	`$486` kann `G59.1`–`G59.3` sperren.
`G61`	Exakte Bahn	
`G61.1`	Genauhalt	Die Achsen halten am Ende jeder programmierten Bewegung an.
`G64 P Q`	Bahnverschleifung	Wo es Zeit spart, wird die Ecke zwischen zwei geraden Bewegungen zu einem Bogen innerhalb des Abstands P von der Ecke. Q wird akzeptiert und nicht verwendet.
`G65 P1`–`P6`	Eingebaute Makros	Einstellungen, Werkzeugkorrekturen, Parameter, Maschinenzustand, Wahl des Messtasters, Warten auf die Spindel. `P7` (Modbus-Nachrichten) wird nicht unterstützt.
`G65 P100` und höher, `G66`, `G66.1`, `G67`	Dateimakros, modale Makroaufrufe	P muss eine ganze Zahl sein. Kein anderer Befehl darf im selben Satz stehen (`error:21`).
`G73`, `G81`, `G82`, `G83`	Bohrzyklen, Tieflochbohren mit Entspanen	Jede Bohrung erhält die Bohrschritte, die sie braucht.
`G84`, `G85`, `G86`, `G89`	Gewindebohr- und Ausbohrzyklen	
`G80`	Zyklus aus	
`G90`, `G91`	Absolute Maße, inkrementelle Maße	
`G91.1`	Inkrementelle Bogenmittelpunkte	

Code	Bedeutung	Hinweise
`G92`, `G92.1`, `G92.2`, `G92.3`	Koordinatenverschiebung: setzen, löschen, aussetzen, wiederherstellen	
`G93`, `G94`, `G95`	Inverse Zeit, Einheiten pro Minute, Einheiten pro Umdrehung	Kreisbögen in inverser Zeit benötigen die programmierte Zeit.
`G96`, `G97`	Konstante Schnittgeschwindigkeit, konstante Spindeldrehzahl	Nur im Drehmodus.
`G98`, `G99`	Zyklusrückzug auf die Starthöhe, auf die R-Ebene	

Diese G-Codes werden nicht unterstützt (`error:20`):

- `G5.2`, `G5.3` (NURBS);
- `G17.1`–`G19.1` (UVW-Ebenen);
- `G70`–`G72` (Drehzyklen);
- `G90.1`.

M-Code

Code	Bedeutung	Hinweise
`M0`, `M1`	Programmpause, wahlweiser Halt	Der Echtzeitbefehl `0x88` schaltet den wahlweisen Halt (`M1`) ein oder aus.
`M2`, `M30`	Programmende	
`M3`, `M4`, `M5`	Spindel im Uhrzeigersinn, gegen den Uhrzeigersinn, aus	
`M6`	Werkzeugwechsel	`$341` legt fest, was M6 tut. M6 und eine Bewegung in einem Satz werden mit `error:40` beantwortet.
`M7`, `M8`, `M9`	Kühlung: Sprühkühlung, Flutkühlung, aus	Bei der MC-1 betreibt M8 die Wasserkühlung der Spindel.
`M25`	Z hoch auf die Referenzhöhe	Centroids Code, den manche CAM-Postprozessoren ausgeben.
`M48`, `M49`	Overrides ein, aus	
`M50`–`M53`	Vorschub-Override, Spindel-Override, adaptiver Vorschub, Vorschub halt: ein oder aus	
`M56`	Parken ein oder aus	
`M60`	Pause für den Palettenwechsel	
`M61`	Das aktuelle Werkzeug setzen, ohne zu wechseln	
`M62`–`M65`	Digitalausgang ein oder aus, mit der nächsten Bewegung oder sofort	Auf Platinen mit Zusatzausgängen.
`M66`	Auf einen Eingang warten	Das Warten beginnt, wenn die Bewegungen davor beendet sind.
`M70`–`M73`	Den modalen Zustand sichern und wiederherstellen	
`M98 P L`, `M99`	Unterprogramm aufrufen, zurückkehren	In Programmen, die aus dem Dateisystem der Steuerung laufen. Ein Satz für Satz gesendetes `M98` wird mit `error:39` beantwortet. Ein `M99` im Hauptprogramm beendet es.

Diese M-Codes werden nicht unterstützt (`error:20`):

- `M19` (Spindelorientierung);
- `M67`, `M68` (Analogausgänge; die MC-1 hat keine);
- `M100`–`M199` (benutzerdefinierte M-Codes).

Wörter, Syntax und Parameter

Aufbau eines Satzes

- Kommentare `( … )` und `;` sind erlaubt.
- `(MSG,…)` sendet den Text als Meldung an das Sender-Programm.
- `(PRINT,…)` sendet eine Meldung mit den Werten der Parameter, die es nennt.
- `(DEBUG,…)` tut dasselbe, solange `#5599` eingeschaltet ist.
- Satzausblendung `/`, Satznummern `N` und `%`-Zeilen (mit oder ohne Namen) sind erlaubt.
- Groß- und Kleinschreibung spielt keine Rolle, und Leerzeichen innerhalb von Wörtern werden ignoriert.

Parameter und Ausdrücke

- Nummerierte Parameter `#1`–`#5000` gehören zum Programm. Sie haben den Wert 0, bis das Programm sie setzt.
- `#5001` und höher sowie die benannten Systemparameter sind nur lesbar.
- Benannte Parameter `#<name>`: Das Lesen eines Namens, der nie gesetzt wurde, wird mit `error:39` beantwortet. `EXISTS[#<name>]` zeigt, ob er gesetzt wurde.
- Ausdrücke in `[ ]` verwenden die Operatoren und Funktionen von LinuxCNC. Winkel werden in Grad angegeben.
- Ein Wert, den ein Satz setzt, kann im selben Satz nicht gelesen werden.

O-Wörter

- Programme, die aus dem Dateisystem der Steuerung laufen, können Unterprogramme sowie `if`, `elseif`, `else`, `while`, `do` und `repeat` verwenden.
- Unterprogramme und Schleifen, die Satz für Satz gesendet werden, werden gesammelt und ausgeführt, sobald sie vollständig sind.
- Ein Unterprogramm kann auch eine eigene Datei sein: `<name>.ngc` oder `<name>.macro`.

Polarkoordinaten

`@` ist der Radius und `^` der Winkel, in der XY-Ebene um den Werkstücknullpunkt.

Fehler in einem Satz

- Zwei Codes derselben modalen Gruppe in einem Satz werden mit `error:21` beantwortet.
- Zwei Befehle in einem Satz, die beide die Achswörter verwenden, werden mit `error:24` beantwortet.
- Die vollständige Liste der Fehlercodes steht in Fehlercodes.

CAM-Strategien und -Parameter

Was jede Strategie tut, was ihre Parameter bedeuten und wie der Zwilling die Schnittdaten festlegt. Die Zahlen, die der Zwilling nennt, sind Schätzungen aus seinen physikalischen Modellen.

Die folgenden Parameter sind die, die ein Formelement in einem Auftrag annimmt (die Formelemente des Tools `program_generate`). Längen sind in Millimetern angegeben, Winkel in Grad.

Eine Tasche ausräumen: Ringe, Zickzack und adaptiv

Die Fläche einer Tasche ist ihre Form abzüglich ihrer Inseln, um den Radius des Werkzeugs nach innen versetzt. Sie wird Schicht für Schicht ausgeräumt, jede Schicht nicht tiefer als die Schnitttiefe (`ap`). Die Strategie (`strategy`) legt den Weg in einer Schicht fest:

Strategie	Weg	Geeignet für
`rings` (Standard)	Versatzkonturen der Fläche von ihrer innersten Stelle nach außen bis zur Wand, jeweils `ae` voneinander entfernt	Die meisten Taschen
`zigzag`	Gerade Bahnen hin und her unter dem Winkel `angle`, jeweils `ae` voneinander entfernt	Offene, flache Bereiche
`adaptive`	Ein Weg, der den Eingriff des Werkzeugs überall nahe `ae` hält	Tiefe Taschen, härtere Materialien

Eine Tasche mit einer Passung schlichtet ihre Wände in einem eigenen Gang auf Maß. Siehe Passungen.

Adaptives Ausräumen schneidet mit kleinem radialem Eingriff bei größerer Schnitttiefe `ap`. Seine Verbindungswege verlaufen frei vom Material, und wo die Fläche dafür zu schmal ist, greift es auf Ringe zurück.

Ein Schlichtumlauf umschlingt das Werkzeug in einer Innenecke der Wand viel stärker als an einer geraden Wand. Wo die Umschlingung mehr als das 1,2-Fache beträgt, wird dieser Teil der Wand zuerst in Stufen geschnitten (`corner_steps`, standardmäßig an; aus, wenn das Feature ein eigenes `finish_ap` angibt, außer es gibt auch `corner_steps: true` an; `corner_steps: false` schaltet es ab).

Restmaterialbearbeitung

Mit `rest: true` räumt eine Tasche nur das aus, was die Werkzeuge davor stehen gelassen haben: die Ecken und engen Stellen, die ein größeres Werkzeug nicht erreichen konnte. Studio ermittelt, was übrig ist, aus den früheren Taschen im selben Programm oder aus dem Werkstück des Zwillings, wenn dieser eines hat.

Konturen und Haltestege

Eine Kontur (`side`: `outside`, `inside` oder `on`) folgt einem Umriss Schicht für Schicht, im Gleichlauf. `outside` und `inside` versetzen den Umriss um den Radius des Werkzeugs; `on` folgt ihm, wie er ist.

Haltestege (`tabs`: Anzahl, Breite und Höhe) lassen unten kurze Brücken stehen, damit ein ausgeschnittenes Teil an seinem Platz bleibt. Ein Haltesteg ist 1 mm hoch, sofern Sie keine Höhe angeben; bei einem Schnitt von höchstens 1 mm Tiefe ist er halb so hoch wie die Tiefe.

Nuten: Schichten und trochoidal

Eine Nut (`x0`, `y0`, `x1`, `y1`, `width`) wird mit der Strategie gefräst:

- `layers` (Standard): die Mitte der Nut in voller Werkzeugbreite, in Schichten, dann nach außen bis zur Breite der Nut auf jeder Seite;
- `trochoidal`: kleine Schleifen entlang der Nut, jede um `step` weiter. Das Werkzeug ist nur über einen kurzen Bogen im Eingriff und kann daher in einem Gang tiefer schneiden. Die Nut muss breiter sein als das Werkzeug.

Wie ein Werkzeug ins Material eintaucht: Helix, Rampe, senkrechtes Eintauchen

Ein Schaftfräser taucht nicht ins volle Material ein. Jede Schicht beginnt mit:

- einer Helix, wo die Fläche Platz dafür hat (Durchmesser und Winkel stehen im `entry` der Antwort);
- einer Rampe entlang des Wegs, wo das nicht der Fall ist;
- einem senkrechten Eintauchen nur in eine Bohrung oder außerhalb des Rohteils.

Die Antwort nennt für jede Bearbeitung, wie sie zuerst ins Material eingetaucht ist.

Bohrungen

Bohrzyklen

Bohrungen an `points`, `depth` tief, werden mit den Zyklen der Steuerung gebohrt. Eine Bohrung, die tiefer ist als der dreifache Bohrerdurchmesser, wird mit Entspanen gebohrt, mit dem 1,5-fachen Durchmesser je Bohrschritt, sofern `peck` keinen anderen angibt. Die Rückzugsebene des Zyklus liegt 1 mm über der Oberseite.

Ausbohren durch Helixinterpolation

Eine Bohrung mit dem Durchmesser `d`, die größer ist als der Schaftfräser, wird auf einer Helix gefräst und dann am Boden auf Maß geschlichtet. Die Bohrung darf höchstens doppelt so groß sein wie der Durchmesser des Werkzeugs; größere Bohrungen werden als Tasche gefräst (`shape: circle`).

Reiben

Eine Reibahle (`reamer 6 H7`) mit der Passung der Bohrung fährt mit ihrem Vorschub ein und mit demselben Vorschub wieder aus, ohne Eilgang aus der Bohrung. Die Bohrung muss davor im selben Auftrag gebohrt werden, 0,05–0,6 mm kleiner, oder als `predrilled` gekennzeichnet sein. Die Drehzahl der Reibahle beträgt ein Viertel der Schnittgeschwindigkeit des Schaftfräsers für das Material, ihr Vorschub pro Umdrehung das 2,5-Fache des Bohrervorschubs.

Gewindefräsen, auch kegelige Rohrgewinde

Ein Gewindefräser fährt die Bohrung hinab und schneidet das Gewinde dann von unten nach oben auf einer Helix mit einer Steigung pro Umdrehung: Rechtsgewinde gegen den Uhrzeigersinn (Gleichlauf), Linksgewinde im Uhrzeigersinn. Das Gewinde wird benannt (`thread`: `M8`, `M8x1`, `1/4-20 UNC`, `G1/4`) oder durch `d` und `pitch` angegeben. Kegelige Rohrgewinde (`1/4 NPT`, `Rc1/4`) werden auf ihrem Kegel geschnitten, jede Windung in 48 Bögen.

Ringnuten in einer Bohrung

`groove` fräst mit einem T-Nutenfräser (einer Scheibe auf einem dünneren Hals) eine Ringnut in die Wand einer Bohrung (für einen Sicherungsring oder einen O-Ring, oder eine Lichtfalle). Der Fräser fährt in der Mitte der Bohrung bis zur Nut hinab, fährt bis 0,1 mm vor die Wand und umrundet für jedes `step` in die Wand eine volle Umdrehung (Gleichlauf), dann zurück zur Mitte und nach oben. Der Auftrag gibt den Nutdurchmesser `d`, den Bohrungsdurchmesser `bore`, die Unterkante der Nut `depth` und ihre Höhe `width` an (standardmäßig die Dicke der Scheibe; eine höhere Nut wird von unten nach oben in mehreren Durchgängen gefräst). Abgelehnt wird, wenn das Werkzeug in der Spindel kein T-Nutenfräser ist, wenn die Scheibe nicht mit 0,2 mm Spiel durch die Bohrung passt, wenn die Nut tiefer ist, als die Scheibe über ihren Hals hinausreicht, und wenn die Nut nicht größer als die Bohrung ist.

Fasen, Gravieren und V-Gravur

- Fase: Ein V-Fräser folgt der Kante des Umrisses in der Tiefe, die die `width` ergibt. Die Breite wird durch den Kegel begrenzt.
- Gravur: Das Werkzeug folgt Linien oder Text in einer Einstrich- oder einer Konturschrift in einer festen `depth`.
- V-Gravur: Der V-Fräser folgt der Medialachse der Form. Seine Tiefe an jedem Punkt ergibt die Breite dort, sodass die Ecken der Form scharf werden. `max_depth` macht breite Bereiche in dieser Tiefe flach.

Zum Gravieren auf einer gekrümmten Fläche geben Sie `surface` an: den Namen eines Netzes aus `geometry_import`. Jeder Strich wird alle 0,2 mm abgetastet, das Werkzeug wird auf das Netz gesetzt und geht `depth` unter die Stelle, an der es aufsetzt; an einer Schräge berührt ein V-Fräser zuerst auf der höheren Seite, daher wird nirgends tiefer als `depth` geschnitten. Ein Strich außerhalb des Netzes wird abgelehnt.

Perlage, das Muster sich überlappender Kreise der Uhrmacher, ist `perlage`: Ein Schaftfräser so breit wie ein Kreis fährt an jedem Punkt eines Rasters innerhalb einer Kontur auf `depth` hinab (höchstens 0,2 mm), wartet `dwell` Sekunden (standardmäßig 0,2) und hebt 0,3 mm über die Fläche ab, um zum nächsten zu fahren. Die Reihen sind um eine halbe `pitch` versetzt (standardmäßig das 0,65-Fache des Werkzeugdurchmessers), jeder Kreis überdeckt einen Teil des vorigen. `margin` hält die Kreismittelpunkte so weit innerhalb der Kontur.

3D-Schruppen und -Schlichten

Ein Netz wird in Ebenen geschruppt (`rough3d`), jede Ebene adaptiv ausgeräumt. Danach wird es geschlichtet (`finish3d`) mit:

Strategie	Weg
`parallel`	Bahnen in einer Richtung über das Teil
`waterline`	Konturen auf gleicher Höhe, für steile Wände
`steep_shallow` (Standard)	Z-konstant auf den steilen Bereichen, parallel auf den flachen
`pencil`	Entlang der Innenecken, nach den anderen

`rest_from` schlichtet nur, was ein früheres Werkzeug stehen gelassen hat.

Restschruppen: Mit `rest_from` (der größere Schaft- oder Torusfräser, der zuvor geschruppt hat, als Angabe oder Werkzeugnummer) schneidet `rough3d` nur, was dieses Werkzeug stehen ließ, etwa die Lücken zwischen Rippen, in die es nicht kam; `rest_allowance` ist das Aufmaß, das es ließ (standardmäßig das dieses Durchgangs).

Schnittdaten aus dem Zwilling

Studio geht von den Daten des Materials für die Art des Werkzeugs aus: dem Bereich der Schnittgeschwindigkeit und dem Vorschub pro Zahn. Werte, die Sie in `cutting` angeben (`rpm`, `feed`, `ap`, `ae`, `finish_ap`), bleiben so, wie Sie sie angeben. Die übrigen Werte werden mit den Modellen des Zwillings festgelegt.

Stabilität und Rattern

Das Rattermodell des Zwillings berechnet die Stabilitätskarte für das Werkzeug in seinem Halter: die Schnitttiefe, die bei jeder Spindeldrehzahl stabil bleibt. Die stabilen Bereiche liegen nur dort, wo das Modell von Werkzeug und Maschine sie hinlegt; deshalb zielt Studio nicht auf den höchsten Punkt eines Bereichs: Für jede Drehzahl nimmt es die kleinste stabile Tiefe innerhalb von 10 % um sie, wählt eine Drehzahl zwischen dem 0,7- und dem 1,8-Fachen der Drehzahl, die das Material vorgibt, und schneidet höchstens 80 % dieser Tiefe.

Leistung, Kraft und die Durchbiegung des Werkzeugs

- Leistung: Der Schnitt nimmt bei dieser Drehzahl höchstens etwa 80 % der Leistung der Spindel in Anspruch.
- Kraft: Die Schnittkraft bleibt unter der Hälfte dessen, was das Spannmittel seitlich hält.
- Durchbiegung: Ein Schlichtgang wird so leicht gehalten, dass die Schätzung des Zwillings für die Durchbiegung des Werkzeugs klein bleibt.

- Ecken und Bögen: Der Vorschub auf einem Bogen wird durch die Beschleunigung begrenzt, die die Maschine hat, damit die Steuerung nicht unter den Vorschub verlangsamt, für den der Schnitt geplant wurde. Schlichtgänge werden in den Innenecken einer Tasche langsamer, wo der Eingriff des Werkzeugs wächst.

Der Regelkreis: das Programm im Zwilling ausprobieren

Bei eingeschalteter Verifizierung führt Studio das Programm auf einem frischen Zwilling aus. Zeigt der Lauf Rattern, eine zu hohe Belastung oder ein Werkzeug, das sich zu stark durchbiegt, ändert Studio das Programm und versucht es erneut:

- flachere Schichten,
- eine andere Spindeldrehzahl zwischen dem 0,7- und dem 1,8-Fachen der eingestellten, bei der der Schnitt stabil bleibt, auch wenn die stabilen Bereiche etwas anders liegen,
- Nachschnitte an den Wänden einer Passung.

Jede ausprobierte Version wird gespeichert. Ein langes Programm wird im Hintergrund des Aufrufs ausprobiert, und sein Bericht kommt später (`run_status`). Die Antwort sagt, was sich geändert hat und warum.

Mehrere Aufspannungen und Verschachtelung

- Aufspannungen: Studio schreibt das Programm jeder Aufspannung und prüft die Aufspannungen dann der Reihe nach im Zwilling, jede an dem Werkstück, das die Aufspannungen davor übrig gelassen haben. Wo sich dieses Werkstück vom Rohteil unterscheidet, wird das Programm der Aufspannung dafür neu geschrieben.
- Verschachtelung: Teile werden anhand ihrer Umrisse auf der Platte platziert, mit einem Abstand voneinander. Jedes Teil behält seine Formelemente, die mit ihm gedreht und verschoben werden.

Grenzen

- Drei Achsen. Keine vierte Achse, keine Werkzeugneigung.
- STEP-Dateien werden nicht gelesen; exportieren Sie STL oder 3MF aus dem CAD-Programm.
- Die Schnittdaten und die Prüfungen des Zwillings sind Schätzungen aus physikalischen Modellen der Maschine, des Werkzeugs und des Materials. Ein Werkzeug oder Material, das von seinem Modell abweicht, schneidet anders.
- Editionen: Adaptives Ausräumen, die 3D-Schlichtstrategien außer `parallel`, die Restmaterialbearbeitung und mehrere Aufspannungen sind ab der Maker-Edition enthalten; die Verschachtelung und der Programmoptimierer sind in der Pro-Edition enthalten.

Wie groß ein Auftrag sein kann

Jenseits dieser Grenzen schreibt Studio das Programm nicht. Es sagt, warum, und was zu ändern ist.

Was	Grenze
Aufspannungen in einem Auftrag	8
Verschachtelung	16 Teilearten, je 200 Exemplare
Ein Netz (STL, 3MF, OBJ, glTF)	5 Millionen Dreiecke; die Datei bis zu 1 GB
Eine SVG-, DXF- oder Textdatei	64 MB; eine DXF-Zeichnung bis zu 2 Millionen Segmente
Adaptives Ausräumen	Das Raster der Wegberechnung fasst etwa 1,5 Millionen Zellen. Eine Tasche in der Größe des Verfahrwegs der MC-1 lässt sich mit einem 3-mm-Schaftfräser ausräumen; mit einem 1-mm-Fräser verwenden Sie Ringe oder Zickzack oder einen größeren Bahnabstand.

Ein Auftrag kann bis zu 32.000 Formelemente haben; gleiche Bohrungen werden als ein Formelement mit ihren Punkten angegeben. Die Bearbeitungsliste, die Begründungen und die Reparaturen des Regelkreises gelten für die ersten 64 Formelemente; ein späteres Formelement, das rattert, wird gemeldet, aber nicht repariert. Das Einrichteblatt listet die ersten 64 Bearbeitungen auf und sagt das auch.

Die Programmprüfung und das Einrichteblatt lesen bis zu 2 Millionen Bewegungen. Ein Kreisbogen zählt als seine kurzen Sehnen: etwa 50 für einen Vollkreis mit 3 mm Radius, etwa 220 für einen mit 50 mm. Darüber hinaus sagen sie, wo sie aufgehört haben. Das Einrichteblatt gibt weiterhin die Zeit des gesamten Programms an, aber nicht die Zeiten der Bearbeitungen oder die Höhen der Werkzeuge hinter diesem Satz. Der Probelauf im Zwilling prüft das gesamte Programm.

Wie lange Studio zum Schreiben eines Programms braucht, gemessen auf zwei Prozessorkernen eines Desktop-PCs (danach folgt der Probelauf im Zwilling):

Auftrag	Zeit	Speicher
20 Taschen mit Inseln auf einer Platte von 180 x 130 mm	unter 1 s	18 MB
Eine Tasche in der Größe des Verfahrwegs, adaptiv, 6-mm-Schaftfräser	12 s	47 MB
Dasselbe mit einem 3-mm-Schaftfräser	37 s	70 MB
1000 gravierte Zeichen	12 s	25 MB
200 Unterlegscheiben, auf einer Platte verschachtelt	5 s	30 MB
3D-Schruppen eines Reliefs in der Größe des Verfahrwegs (eine halbe Million Dreiecke), 6-mm-Schaftfräser	8 min	240 MB
Dasselbe Relief aus 3 Millionen Dreiecken	20 min	1 GB

Auftrag	Zeit	Speicher
3D-Schlichten desselben Reliefs, parallel, 2-mm-Kugelfräser, 0,01 mm Rautiefe (eine halbe Million Dreiecke)	4 min	270 MB
Dasselbe aus 3 Millionen Dreiecken	18 min	1 GB

Für 3D-Arbeiten:

- Machen Sie das Netz nur so fein, wie das Teil es braucht. Das Schlichten dauert ungefähr in dem Maß länger, in dem das Netz mehr Dreiecke hat.
- Schlichten Sie eine sanft gewellte Fläche mit `parallel` oder `steep_shallow`, nicht mit `waterline`. Die Strategie Z-konstant macht dort viele kleine Schleifen: Auf dem Relief oben bewegte sie sich mehr in der Luft als im Material, und ihr Programm hatte 2,8 Millionen Sätze.

Passungen, Rundlauffehler und Prüfmethoden

Wie Studio ein Maß auf eine Passung fräst, wie es den Rundlauffehler des Fräsers berücksichtigt und wie es ein Teil mit dem Messtaster misst. Die Zahlen des Zwillings sind hier Schätzungen aus seinen physikalischen Modellen; Studio erhebt keinen Anspruch auf eine Positioniergenauigkeit der Maschine.

Passungen (ISO 286)

Ein Maß kann eine Passung tragen: den Durchmesser einer Bohrung (`"d": "Ø12 H7"`), den einer Welle (`"fit": "g6"` an einer Außenkontur), die Breite einer Nut oder die X- und Y-Maße eines Rechtecks. Studio entnimmt die Grenzabmaße der Passung für das Maß den Tabellen der ISO 286-2 oder verwendet die Grenzabmaße, wie Sie sie schreiben (`"tol": "+0.015/0"`, in Millimetern).

Auf die Mitte des Toleranzfeldes fräsen

Die Wand wird mit einem Aufmaß geschruppt und dann in einem eigenen Gang auf die Mitte des Toleranzfeldes der Passung geschlichtet. Passungen mit dem Toleranzgrad IT6 und feiner erhalten einen Nachschnitt: den Schlichtumlauf noch einmal, auf Maß, um abzunehmen, was die Durchbiegung des Werkzeugs übrig gelassen hat. Der Regelkreis fügt Nachschnitte dort hinzu, wo die Schätzung des Zwillings für die Durchbiegung sie verlangt.

Die Antwort nennt für jede Passung das Toleranzfeld, den Durchmesser, auf den sie gefräst wurde, und warum.

Gebräuchliche Passungen

Name	Paarung	Verwendung
`sliding`	H7/g6	Dreht oder gleitet, kleines Spiel
`locating`	H7/h6	Die engste Spielpassung: von Hand gefügt
`transition`	H7/k6	Etwas Spiel oder etwas Übermaß: Zentrieren, leichtes Pressen
`press`	H7/p6	Übermaß: eingepresst

Eine Paarung nennt beide Teile: Die Bohrung erhält die erste Angabe, die Welle die zweite.

Der Rundlauffehler des Fräsers

Ein Fräser mit Rundlauffehler schneidet eine Bohrung, die größer ist als er selbst. Der Zwilling modelliert den Rundlauffehler des Werkzeugs in seinem Halter als Hüllkurve. Bei einem Maß mit Passung verschiebt Studio den Schlichtweg um diese Hüllkurve, bei einer Bohrung nach innen, bei einer Welle nach außen, sodass das Maß, das der Zwilling schätzt, weiterhin die Mitte des Toleranzfeldes ist. Eine geriebene Bohrung wird nicht verändert: Sie hat das Maß der Reibahle.

Die Hüllkurve ist die Schätzung des Zwillings für sein Modell von Werkzeug und Halter. Messen Sie das erste Teil.

Messen mit dem Messtaster

Eine Antastung und ihre Kompensation

Jede Antastung ist eine Folge von Tastbewegungen (G38):

1. eine geschützte Bewegung bis nahe an die Oberfläche, die anhält, wenn der Messtaster unterwegs auslöst;
2. ein schnelles Antasten der Oberfläche;
3. eine Rückzugsbewegung, bis der Messtaster freigibt, dann noch ein wenig weiter;
4. ein langsames Antasten der Oberfläche: Nur dessen Auslösepunkt wird verwendet;
5. eine Rückzugsbewegung dorthin, wo sie begann.

Der Auslösepunkt ist der Mittelpunkt der Kugel. Studio addiert den wirksamen Radius der Kugel in Richtung der Antastung. Der wirksame Radius und das Lobing des Messtasters stammen aus seiner Kalibrierung an einer Ringlehre mit bekanntem Durchmesser.

Bohrungen, Zapfen, Wände und Böden

- Bohrungen und Zapfen: vier oder acht Antastungen rund um die Wand in einer Tiefe. Studio legt nach der Methode der kleinsten Quadrate einen Ausgleichskreis durch die Kugelmittelpunkte und addiert dann den wirksamen Radius.
- Wände: Antastungen senkrecht zur Wand, an Punkten entlang der Wand.
- Böden und Oberseiten: Antastungen senkrecht nach unten.
- Positionen werden in den Koordinaten des Programms angegeben, vom Werkstücknullpunkt aus.

Allgemeintoleranzen (ISO 2768-1)

Ein Maß ohne eigene Toleranz erhält die Allgemeintoleranz der verlangten Klasse (f, m, c, v) aus der Tabelle der ISO 2768-1 für seinen Maßbereich: Längenmaße sowie die Höhen und Winkel von Fasen.

Ist das Raster des Zwillings für eine Toleranz zu grob, wird die Schätzung des Zwillings für diese Prüfung als zu grob zur Beurteilung gemeldet; der gemessene Wert wird dennoch beurteilt.

Formprüfungen (ISO 2768-2, erster Schritt)

Die Klasse H, K oder L nach der Klasse der ISO 2768-1 (zum Beispiel „mK“) fügt Formprüfungen hinzu, die nach den Tabellen der ISO 2768-2 beurteilt werden:

Prüfung	Antastungen	Tabelle
Ebenheit	Ein Boden auf einem 3 × 3-Raster; die Streubreite der Punkte um ihre Ebene	1
Geradheit	Eine lange Wand, 3 bis 8 Punkte entlang der Wand; die Streubreite um ihre Gerade	1
Rechtwinkligkeit	Eine Wand in einer zweiten Tiefe; ihre Neigung über die Höhe der Wand, bezogen auf Z der Maschine	2
Rundheit	Eine Bohrung oder ein Zapfen an 8 Punkten; die Streubreite der Radien	4 (die Toleranz des Durchmessers)

Zwei Antastungen an einer Wand lassen sich nur unterscheiden, wenn sie weit genug voneinander entfernt sind. Eine Wand, die dafür zu kurz ist, zum Beispiel unterhalb der Fase an ihrer Kante, wird mit der Begründung statt mit einem Wert aufgeführt.

Ein Messtaster, der nur auf seinen wirksamen Radius kalibriert ist, hat ein eigenes Lobing. Es zeigt sich in der Rundheit: Der Bericht nennt, wie groß das Messtastermodell des Zwillings es annimmt.

Fasen und V-Nuten

- Eine Fase wird zweimal entlang ihrer Schräge angetastet, bei 0,3 und 0,7 ihrer Breite vom Umriss. Die Gerade durch die beiden Kugelmittelpunkte verläuft parallel zur Schräge: Sie ergibt Breite, Tiefe und Winkel der Fase.
- Eine V-Nut (eine Gravur oder eine V-Gravur) wird auf jeder Flanke zweimal angetastet. Die beiden Geraden ergeben die Tiefe der Nut an ihrer Spitze, ihre Breite oben, ihren Öffnungswinkel und ihren Versatz gegenüber der Linie, entlang der sie gefräst wurde.
- Ein Gewinde wird nicht gemessen. Seine Spitzen bilden die gerade Bohrung, und der Messtaster des Zwillings trifft nur die Oberseiten der Zellen; die Antwort sagt das.

Was die Schätzung des Zwillings bedeutet

Der Zwilling schneidet sein simuliertes Werkstück mit demselben Programm und führt dann dieselben Antastungen daran mit demselben Messtastermodell aus. Seine Schätzung ist das, was diese Berechnung ergibt. Sie zeigt die eigene Wirkung des Programms auf jedes Maß, soweit der Zwilling sie modelliert: die Durchbiegung des Werkzeugs, seinen Rundlauffehler, die Schnittkräfte.

Die Schätzung ist nicht die Messunsicherheit der Maschine. Der Bericht sagt nichts darüber, wie genau die Maschine misst oder positioniert.

Grenzen

- Der Messtaster misst, was er berührt. Ein Formelement, das schmaler ist als die Kugel, oder eine Fläche, die die Kugel nicht senkrecht erreichen kann, wird nicht gemessen; die Antwort sagt, warum.
- Das Werkstück des Zwillings ist ein Raster. Seine Schätzungen sind nur so fein wie dieses Raster; wo es für eine Toleranz zu grob ist, wird die Schätzung des Zwillings nicht beurteilt.
- Die Formprüfungen sind ein erster Schritt der ISO 2768-2: Ebenheit, Geradheit, Rechtwinkligkeit und Rundheit. Symmetrie und Parallelität werden noch nicht geprüft.

Maschinenprofile

Entwurf (Runde 14): Der Text wird noch geschrieben; die Abschnitte können noch wachsen.

Was ein Maschinenprofil ist

Ein Maschinenprofil ist die Datei, aus der der Zwilling seine Maschine aufbaut: eine Datei pro Maschinenmodell, in JSON. Sie enthält die Geometrie der Maschine und ihre Achsen, die Antriebe und die Kugelgewindetriebe, die Spindel und ihren Antrieb, die Sensoren und ihre Verdrahtung, die Spannmittel, die Schnittdaten für jedes Material, den Werkzeuglängensensor und den Messtaster, die Geräusche und das, womit die Steuerung gebaut wird. Die Einstellungen der Steuerung gehören als Einstellungsdatei dazu.

Das Profil hält fest, woher seine Werte stammen (seine Herkunft): aus der Konstruktion, einem Datenblatt, einer veröffentlichten Quelle, einer Annahme oder einer Messung.

Woher es kommt

Das Profil der MC-1 wird aus der parametrischen Konstruktion der Maschine erzeugt: Dieselben Parameter, die ihre Zeichnungen und ihre Stückliste erzeugen, erzeugen auch ihr Profil, sodass Zwilling und Konstruktion der Maschine nicht auseinanderlaufen können. Das Profil behält Prüfsummen der Konstruktionsdateien, aus denen es erzeugt wurde. Ein Schema prüft seine Form; das Referenzkapitel Schema des Maschinenprofils listet jeden Schlüssel auf.

Die Werte Ihrer Maschine

Jede MC-1 soll bei der Inbetriebnahme vermessen und ihr Zwilling auf die Messwerte eingestellt werden (geplant; das Verfahren und seine Werkzeuge sind geschrieben und am Zwilling getestet):

- die Endschalter: wo sie schalten, ihre Hysterese und Wiederholgenauigkeit;
- jede Achse nach ISO 230-2: Umkehrspiel, die Tabelle der Steigungsfehler, die Schritte pro Millimeter;
- die Schnittkoeffizienten aus Testnuten;
- Trägheit und Reibung der Spindel aus einem Auslaufversuch;
- der Messtaster an einer Ringlehre.

Die gemessenen Werte werden mit Datum, mit dem Werkzeug, das sie gemessen hat, und mit der Art der Messung aufgezeichnet; das Profil wird mit ihnen neu aufgebaut, jeder Wert als gemessen markiert und der Wert der Konstruktion in seiner Historie behalten, und die Einstellungen der Steuerung, die sie ändern, werden an die Maschine gesendet.

Mehrere Maschinen

Studio kann mehrere Maschinen eines Modells führen. Jede behält zwischen den Sitzungen ihre eigene Welt (wo sie steht, ihr Werkstück, ihre Werkzeuge, den Speicher ihrer Steuerung) und ihr eigenes Fensterlayout; das Profil des Maschinenmodells ist für alle dasselbe, bis eine Maschine vermessen wird.

Die Referenz

Jeder Schlüssel des Profils mit seinem Typ, seiner Einheit und dem Wert der MC-1: Schema des Maschinenprofils.

Vorschau: Die Sicherheitshinweise dieser Seite sind noch nicht muttersprachlich geprüft.

MC-1-Hardware

Entwurf (Runde 14): Der Text wird noch geschrieben; die Abschnitte können noch wachsen.

Die MC-1, wie sie konstruiert ist: Die Angaben in diesem Kapitel sind Konstruktionswerte. Jede gebaute Maschine soll bei der Inbetriebnahme vermessen und ihr Zwilling auf das Gemessene eingestellt werden (siehe Maschinenprofile).

Die Maschine

Die MC-1 ist eine dreiachsige Tischfräsmaschine für Aluminium, Messing und technische Kunststoffe. Ihr Portal ist feststehend; der Tisch bewegt sich in Y, der Schlitten des Portals in X, die Spindel in Z.

Verfahrwege X / Y / Z	200 / 150 / 80 mm
Tisch	300 × 220 × 35 mm, M6-Gewindebohrungen im 25-mm-Raster
Abmessungen, Breite × Tiefe × Höhe	etwa 508 × 490 × 564 mm, mit Motoren, Energieketten und Stellfüßen, ohne Schaltkasten
Masse	etwa 57 kg, ohne Schaltkasten
Rahmen	Aluminiumplatten 6061-T6, verschraubt
Linearführungen	HIWIN-Schienen HGR15 (6), Flanschwagen HGW15CC (12)
Kugelgewindetriebe	SFU1204 (12 mm, 4 mm Steigung), C7, in Lagereinheiten BK10 / BF10
Motoren	NEMA-23-Closed-Loop-Schrittmotoren, 1,2–1,3 N·m (3), mit Klauenkupplungen
Spindel	800 W, wassergekühlt, Ø65 × 195 mm, Spannzange ER11
Schaltkasten	ein eigener Kasten neben der Maschine, auf ihrer rechten Seite

Achsen und Antriebe

Jede Achse ist ein Closed-Loop-Schrittmotor, der seinen Kugelgewindetrieb direkt über eine Klauenkupplung antreibt. Die Antriebe melden eine Störung auf einer Leitung: Die Alarmausgänge der drei Antriebe sind in Reihe an den Eingang für Motorstörung der Steuerung angeschlossen, sodass ein Antrieb, der eine Störung hat oder seine Spannungsversorgung verliert, die Maschine mit einem Alarm stoppt.

Die schnellsten Bewegungen der Konstruktion sind 3000 mm/min in X und Y und 1500 mm/min in Z.

Endschalter und Referenzfahrt

Fünf induktive M8-Schalter markieren die Enden der Verfahrwege: je einer an jedem Ende von X und von Y, jeder auf einem eigenen Eingang, und einer am Z-Schlitten für beide Enden von Z. Die Schalter sind im Ruhezustand geschlossen (Öffner): Ein Drahtbruch oder ein lockerer Stecker wird als ausgelöster Schalter gelesen, sodass die Steuerung einen Alarm auslöst und die Referenzfahrt verweigert, statt in den Endanschlag zu fahren.

Die Referenzfahrt läuft zuerst in Z, dann in X und Y, jeweils zu ihrem positiven Ende (X+, Y+, Z+). Die Maschine muss nach dem Einschalten referenziert werden.

Z und die Schwerkraft

Der Z-Schlitten trägt die Spindel, etwa 6,35 kg. Solange die Antriebe mit Spannung versorgt sind, halten sie Z. Sind sie nicht mit Spannung versorgt (nach einem Not-Halt, einer Antriebsstörung oder einem Stromausfall), kann Z mit der Spindel langsam absinken.

> ⚠ **Warnung**
>
> Referenzieren Sie die Maschine nach einem Not-Halt oder einer Antriebsstörung erneut, bevor Sie fortfahren. Greifen Sie nicht unter die Spindel, solange die Antriebe ausgeschaltet sind.

Zur Inbetriebnahme gehört ein Test bei ausgeschalteten Antrieben: Sinkt Z ab, erhält die Maschine einen Z-Motor mit Bremse oder ein Gegengewicht.

Spindel

Die Spindel ist eine wassergekühlte 800-W-Motorspindel, Ø65 × 195 mm mit Spannzange ER11, zweipolig, bis zu 24.000 min^{-1}. Ein Spindelantrieb Huanyang HY01D523B (Frequenzumrichter, Eingang einphasig 220 V) betreibt sie; die Steuerung stellt seine Drehzahl ein und liest seinen Zustand über RS-485 durch ein galvanisch trennendes Modul. Eine Wasserpumpe kühlt die Spindel.

- Das Störungsrelais des Spindelantriebs ist ebenfalls in die Leitung für Motorstörung geschaltet: Wenn der Spindelantrieb auslöst, stoppt die Steuerung das Programm mit einem Alarm, statt ein stehendes Werkzeug ins Werkstück zu fahren.
- Bei einem Not-Halt verliert der Spindelantrieb zusammen mit den Antrieben seine Spannung, und die Spindel läuft von selbst aus (sie wird nicht gebremst).

Steuerung und Verdrahtung

Die Steuerung ist ein Teensy 4.1 auf einer Trägerplatine T41U5XBB mit optoisolierten Eingängen. Sie führt Twinwrights eigenen Kernel aus, der mit dem seriellen Grbl-1.1-Protokoll kompatibel ist, und ist über USB mit dem Computer verbunden.

Eingang	Was daran angeschlossen ist
Endschalter	X−, X+, Y−, Y+ und Z, fünf Eingänge
Messtaster	der Werkzeuglängensensor und der Messtaster
Not-Halt	der zweite Kontakt des Not-Halt-Tasters
Motorstörung	die Alarmausgänge der drei Antriebe und das Störungsrelais des Spindelantriebs, in Reihe
Zyklusstart, Vorschub halt	die Taster SB2 und SB3 am Schaltkasten

Stromversorgung

Das Netz läuft über einen Leitungsschutzschalter. Ein 24-V-Netzteil ist immer eingeschaltet: Es versorgt die Steuerung, die Sensoren und den Steuerstromkreis, sodass die Steuerung mit dem Computer verbunden bleibt und den Not-Halt meldet. Ein Schütz (KM1) schaltet die 48-V-Versorgung der Antriebe, den Spindelantrieb und die Pumpe.

Die Taster des Schaltkastens

Vier Taster Ø22 an der Tür des Schaltkastens, von vorn gesehen:

	links	rechts
obere Reihe	SB1 Einschalten (grün)	SB0 Not-Halt (rot)
untere Reihe	SB2 Zyklusstart (grün)	SB3 Vorschub halt (gelb)

- **SB1 Einschalten** schließt KM1, das sich danach selbst hält. Nach einem Not-Halt oder einem Stromausfall fährt die Maschine nicht von selbst hoch: Entriegeln Sie den Not-Halt und drücken Sie SB1 erneut.
- **SB3 Vorschub halt** tut, was der Vorschub halt Ihres Sender-Programms (`!`) tut: Die Achsen halten kontrolliert an, die Spindel läuft weiter. Er sitzt unter dem Not-Halt, sodass eine Hand, die den Not-Halt verfehlt, Vorschub halt trifft und nicht Zyklusstart.
- **SB2 Zyklusstart** setzt fort, wie es der Zyklusstart Ihres Sender-Programms (`~`) tut.

> **Achtung**
>
> Vorschub halt ist keine Sicherheitsfunktion. Um die Maschine aus Sicherheitsgründen anzuhalten, verwenden Sie den Not-Halt.

Werkzeuglängensensor und Messtaster

Der Werkzeuglängensensor sitzt hinten links am Tisch, gehalten von zwei M6-Schrauben im Raster; er belegt die Ecke des Arbeitsbereichs bei etwa X 0–57, Y 130–150 mm. Der Werkzeuglängensensor und ein Messtaster teilen sich den Messtastereingang der Steuerung.

Mit Z oben liegt die Spitze eines Werkzeugs, das 15,5 mm aus der Spannzange ragt, 80 mm über dem Tisch und die Spindelnase 130 mm. Die Spitze eines längeren Werkzeugs liegt um dessen zusätzliche Länge tiefer. Ein in einer ER-Spannzange gehaltener Messtaster ist von der Mutter bis zur Messtasterspitze meist 60–90 mm lang: Mit Z oben kommt ein solcher Messtaster nur über niedrigen Aufspannungen frei, bei einem Messtaster von 70 mm etwa 20 mm über dem Tisch.

Studio prüft das für Sie: Wenn Sie das Werkstück einrichten, ein Werkzeug wechseln oder die Werkzeugtabelle bearbeiten, vergleicht es den höchsten Punkt der Aufspannung mit der Spitze jedes Werkzeugs bei Z oben und sagt, welches Werkzeug nicht freikommt und um wie viel.

Werkstückspannung

Die M6-Bohrungen des Tisches im 25-mm-Raster nehmen Spannpratzen, eine Grundplatte oder einen Schraubstock auf. Der Zwilling kennt einen selbstzentrierenden 75-mm-Schraubstock auf einer 10 mm dicken Aluminium-Grundplatte, Parallelunterlagen, Spanneisen, doppelseitiges Klebeband und ein Prisma.

- Mit dem Schraubstock auf seiner Grundplatte liegen die Oberkanten der Backen etwa 66 mm über dem Tisch, sodass ein im Schraubstock gehaltenes Teil bis etwa 71 mm Höhe oder mehr reicht.
- Verwenden Sie für Teile, die Sie antasten wollen, eine niedrige Aufspannung: Spanneisen auf der Grundplatte (die Oberseite des Rohteils 15–30 mm über dem Tisch) oder einen niedrigen Schraubstock. Über einem im 75-mm-Schraubstock gehaltenen Teil hat ein Messtaster keinen Platz.
- Wenn Sie die 10-mm-Grundplatte entfernen, haben Werkzeuge 10 mm mehr Platz; für einen Messtaster genügt das nicht.

Sicherheit

- **Not-Halt (SB0)**: Er öffnet den Spulenstromkreis des Schützes, der die Antriebe, den Spindelantrieb und die Pumpe abschaltet, und meldet es der Steuerung über seinen zweiten Kontakt. Die Achsen und die Spindel werden stromlos; die Spindel läuft von selbst aus; Z kann absinken (siehe Z und die Schwerkraft). Das Entriegeln des Not-Halts startet nichts neu: Drücken Sie SB1, warten Sie, bis die Antriebe anlaufen, setzen Sie die Steuerung zurück und referenzieren Sie.
- **Störungen**: Eine Antriebsstörung, ein Auslösen des Spindelantriebs oder ein Drahtbruch an einem Endschalter stoppt die Maschine mit einem Alarm.
- **Vorschub halt** ist ein kontrollierter Halt für die Arbeit, nicht für die Sicherheit.
- Die MC-1 hat keine Einhausung. Halten Sie Abstand zur Spindel, solange sie sich dreht und solange sich die Achsen bewegen können, tragen Sie einen Augenschutz und bringen Sie Faltenbälge oder Bürstenabdeckungen über den Kugelgewindetrieben in X und Z gegen Späne an.
- Keine der Sicherheitsfunktionen der Maschine verwendet KI, und der KI-Assistent in Studio ist keine Sicherheitsfunktion.

Was bei der Inbetriebnahme gemessen wird

Die obigen Konstruktionswerte sind an jeder gebauten Maschine zu prüfen, und ihr Zwilling ist auf das Gemessene einzustellen:

- der Auslösepunkt jedes Endschalters (induktive Schalter, die seitlich überstrichen werden, können etwa 2 mm später auslösen als gezeichnet), danach der Rückzug nach der Referenzfahrt und die Software-Endschalter;
- die Alarmausgänge der Antriebe: Wird einer abgezogen oder werden die 48 V abgeschaltet, muss das einen Alarm auslösen;
- der Z-Test bei ausgeschalteten Antrieben;
- die Parameter des Spindelantriebs, ein Hochlauf ohne Last am Bedienfeld des Antriebs, danach die Steuerung über RS-485;
- der Schaltkasten: der Not-Halt und SB1 wie oben, jeder Endschalter und der Werkzeuglängensensor einzeln abgesteckt (jeder muss einen Alarm ergeben) und die Taster des Bedienfelds.

Wie der Zwilling die gemessenen Werte übernimmt: Maschinenprofile.

Reference

Diese Referenz wird aus dem Programm erzeugt und ist nur auf Englisch.

MCP tool reference

Twinwright Studio offers its twin to AI clients (Claude, ChatGPT, Cursor and others) over the Model Context Protocol. A client starts the program in the background as `TwinwrightStudio.exe --mcp` (the command is shown under AI > Connect AI clients..., The command and configuration) and calls these 47 tools. Each acts on one machine: the one the user sees in the window (`"machine": "window"`, while the window's AI switch is on) or an invisible practice machine (`"sandbox"`). Parameters marked * are required.

Contents

- Machines: `machine_status`, `machine_control`, `machine_library`, `machine_mdi`
- Set-ups: `setup_workpiece`, `setup_tool`, `setup_work_zero`, `inspect_part`, `inspect_report`, `setup_sheet`, `probe_cycle`, `setup_part`, `setup_copy`, `setup_example`
- Programs: `program_check`, `program_simulate`, `program_run`, `program_load`, `program_debug`, `program_restart`, `program_optimize`, `geometry_import`, `geometry_nest`, `program_generate`, `program_edit`, `program_versions`, `job_get`, `job_edit`
- Runs and results: `run_status`, `run_profile`, `result_compare`, `result_image`, `result_export`, `part_save`, `part_export`, `part_load`, `log_query`
- The licence: `licence_status`
- The part library: `part_list`, `part_get`, `part_update`, `part_delete`
- Reports to support: `report_draft`, `report_list`
- The user's window: `window_show`, `settings`
- The manual: `manual_search`

Machines

`machine_status` - Machine status

read-only · the same call twice does no more than once

The state of a machine: controller state and alarm (with its meaning), homed or not, machine and work positions, G54, the tool in the spindle, the spindle, the workpiece on the table and the program's progress - and the other machines, a line each. detail "full" adds the controller's modes, offsets and planner queue; detail "program" gives the loaded program's text in pages. Changes nothing. Tell the user which machine you looked at: only the window's is the one they see.

Parameters

Name	Type	Default	Description
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"
`detail`	string	`"brief"`	(one of `brief`, `full`, `program`)
`offset`	integer	`0`	detail program: the first line (0-based)
`limit`	integer	`400`	

Returns (structured content): `machine`, `notice`, `status`, `machines`, `program`.

`machine_control` - Operate a machine

changes the machine (destructive)

An operator's action on a machine, then its state. home: homes (resets and unlocks first if it is in an alarm; does nothing when already homed - force: true homes again). unlock: $X, when the position is still known. reset: Ctrl-X. hold / resume: feed hold / cycle start. stop: stops the running program. move: to x, y and/or z (mm) in the work (G54) or machine frame, rapid or at feed; moving X/Y it first goes up to the top of Z, where homing left it (safe_z). power_on / power_off: the main switch (and START). estop / estop_release. restart: a practice machine starts afresh (never the window's). The window's clock, as its Simulation > Speed menu sets it: speed (factor 0.5 .. 1000 times real time - from x100 lite: what only the views see is thinned, the results the same - or 0 for the fastest the computer goes); to_end (on: the program computed to its end, not shown frame by frame, started if only loaded; on false stops it there, as stop); pause_sim / resume_sim (the simulation's own pause: virtual time stands). A practice machine runs as fast as it can already. override: the feed (10 .. 200 %), spindle (10 .. 200 %) and rapid (25, 50, 100 %) overrides, as the window's knobs set them (the status says them: overrides). On the user's machine the user allows each change in the window; when the window's twin follows a real machine (shadow mode) it takes none of these, stops included - the real machine is stopped at its own controls or in its sender.

Parameters

Name	Type	Default	Description
`action` *	string		(one of `home`, `unlock`, `reset`, `hold`, `resume`, `stop`, `move`, `power_on`, `power_off`, `estop`, `estop_release`, `restart`, `speed`, `to_end`, `pause_sim`, `resume_sim`, `override`)
`feed`	number		mm/min (G1); omitted: rapid (G0)

Name	Type	Default	Description
`spindle`	number		override: the spindle speed, % (10 .. 200)
`rapid`	number		override: the rapids, % (one of `25`, `50`, `100`)
`factor`	number		speed: times real time (0.5 .. 1000; 0 the fastest)
`on`	boolean	`true`	to_end: start (true) or stop (false)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"
`x`	number		
`y`	number		
`z`	number		
`frame`	string	`"work"`	(one of `work`, `machine`)
`safe_z`	boolean	`true`	
`force`	boolean	`false`	
`drive`	string		restart: the drive model (L0 ideal motion, fastest; L1 closed-loop drives, the default; L2 flexible transmission with backlash; L3 electrical) (one of `L0`, `L1`, `L2`, `L3`)

Returns (structured content): `machine`, `notice`, `status`, `action`, `did`, `next`.

`machine_library` - The window's machine library

changes the machine

The user's machines in the Studio window's library (several machines of a model, each with its own world - where it stands, its workpiece, tools, controller - and its own interface): list them (which one the window runs); new (name: from the factory state), copy (id, name: its world copied - it starts where that one stands), rename (id, name) - written at once, the window's library page follows; switch (id): the window saves its machine and starts again with that one, with the user's yes in the window (without a window open: the machine it starts with next).

Parameters

Name	Type	Default	Description
`action`	string	`"list"`	(one of `list`, `new`, `copy`, `rename`, `switch`)
`id`	string		copy, rename, switch: the machine (list gives the ids)
`name`	string		new, copy, rename: what the user will call it

Returns (structured content): `machine`, `notice`, `action`, `entry`, `machines`.

`machine_mdi` - Send MDI lines

changes the machine (destructive)

Send G-code or $ lines to a machine as an operator types them: one at a time, each with the controller's answer - ok, or the error and what it means. After a refused line the rest are not sent (the controller would refuse them too) and the result says so. Waits until the machine is idle again and gives its state. For homing, unlocking, stopping or moving, machine_control is simpler.

Parameters

Name	Type	Default	Description
`lines` *	array of string		
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"
`timeout_s`	number	`120`	simulated seconds a line may take

Returns (structured content): `machine`, `notice`, `status`, `lines`.

Set-ups

`setup_workpiece` - Place a blank

changes the machine (destructive)

Put a blank on a machine's table (replacing the one there), take it off (op "remove") or turn it over where it stands (op "flip": the other side up, what was machined stays - for a part's second side; refused while a program runs). Table coordinates in mm: X 0..200 left to right, Y 0..150 front to back, centre [100, 75]. Homes first if needed. The twin refuses setups that do not fit and says why. A part cut through: fixture tape (a thin sheet) or clamps on a sacrificial plate (sacrificial) - the blank that much higher, the cuts through it go into the plate, not the table. Clamps hold the blank's two ends unless sides says otherwise. The window's setup file: op "save" writes the setup on the machine to path (.json, not over a file unless overwrite), op "open" sets it up from one.

Parameters

Name	Type	Default	Description
`op`	string	`"place"`	(one of `place`, `remove`, `flip`, `save`, `open`)
`path`	string		op save / open: the setup file (.json, one line - the window's Setup > Save setup... / Open setup...)
`overwrite`	boolean	`false`	
`turn`	string	`"u"`	op flip: u about the blank's u axis (its length as placed, X at angle 0): front and back change places; v about its v axis: left and right change places (the blank also turns 180 degrees about Z) (one of `u`, `v`)
`shape`	string	`"block"`	(one of `block`, `cylinder`, `tube`, `hex`, `rod`, `stl`)
`size`	array of number		block: [x, y, z] mm; cylinder/rod: [diameter, length]; tube: [outer, inner, length]; hex: [across flats, length]
`material`	string	`"6061-T6"`	(one of `6061-T6`, `7075-T6`, `ADC12`, `HPb59-1`, `H59`, `T2`, `Q235`, `45`, `304`, `HT200`, `POM`, `PMMA`, `PA6`, `HDPE`, `MDF`, `hardwood`, `wax`)
`fixture`	string	`"vise"`	(one of `vise`, `clamps`, `tape`, `fixed`, `vblock`, `none`)
`sides`	string	`"u"`	fixture clamps: the edges the strap clamps hold - u at the blank's two ends along its u axis (its length as placed, X at angle 0), v on its two sides along it, all on all four (one of `u`, `v`, `all`)
`vise`	string	`"precision"`	fixture vise: which - precision (75 mm jaws on a 10 mm subplate, their tops 66 mm over the table) or low (the low-profile vise bolted to the table's grid, jaw tops 20 mm over it: more of the Z travel left for a long tool or a probe, a little less hold) (one of `precision`, `low`)
`protrude`	number		fixture vise: how far the blank's top stands over the jaws, mm (5 if left out; the twin puts it on the jaws' step or on parallels for that, 2 mm gripped at least) - less for a thin plate, e.g. 3 for 6 mm in the low vise
`at`	array of number	`[100, 75]`	blank centre on the table [x, y] mm
`angle`	number	`0`	rotation about Z, degrees
`res`	number		simulation cell size in mm (0.01..0.2; smaller is finer and slower)
`file`	string		shape stl: the blank's STL on this computer
`keep_zero`	string		set G54 on the new blank at once, at this corner (as setup_work_zero does); left out, G54 stays where it was (one of `front-left-top`, `center-top`)
`sacrificial`	object		fixture tape or clamps: the blank on a sacrificial plate, so that what is cut through goes into the plate, not the table. The blank sits higher by the plate (and the tape); the tool may cut into the plate (the summary's setup.sacrificial_cut_mm says how deep), the collet and the spindle may not; the tape holds as on the table, the clamps hold the blank and the plate down together (their tapped holes clear of the plate)
`sacrificial.material`	string	`"MDF"`	a material of the list above (for the drawing and the set-up sheet)
`sacrificial.thickness`	number	`10`	mm, 3 to 25
`sacrificial.size`	array of number		[length, width] mm along the blank's own axes; left out: the blank's footprint and 10 mm each side
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `placed`, `flipped`, `refused`, `reason`, `setup`.

`setup_tool` - Tool and tool table

changes the machine · the same call twice does no more than once

The tool in a machine's spindle (spec) and/or its tool table for programs that change tools with M6 (tools: {"1": spec, ...}; the change and the length measurement then happen by themselves). new: true - a new tool of that spec: its wear book (an estimate, machine_status's tool_wear) starts again. Tool spec: kind and diameter in mm, then options. Kinds: 'flat D', 'ball D', 'bull D RC' (corner radius), 'v D ANGLE' (V-bit), 'drill D ANGLE', 'tslot D THICKNESS dNECK', 'dovetail D ANGLE dNECK', 'thread D PITCH', 'probe D' (a touch probe with a ball of D; ' typical' a kinematic one: it triggers its pretravel past the touch, scattered with a seed; ' ideal' at the touch, the default). Options: 'nN' flutes (default 2), 'LMM' stick-out from the collet (the length reference tool sticks out 15.5 mm), ' sD lR' a shank of diameter D and the reach R below it, ' tirU' the cutter's runout in the collet, TIR in um (a precision collet's class, or as measured; default: the machine's). Examples: 'flat 6 n3 L20', 'ball 3 n2 L16', 'v 6 60 L18', 'flat 1.5 n2 s4 l5 L14', 'flat 0.2 n2 s4 l1 L15 tir3', 'probe 2 L30 typical'.

Parameters

Name	Type	Default	Description
`spec`	string		e.g. 'flat 6 n3 L20'
`new`	boolean	`false`	a new tool: its wear book from 0
`tools`	object of string		the tool table for M6: {"1": "flat 6 n3 L20", ...}
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `tool`, `tool_table`.

`setup_work_zero` - Set G54 on the blank

changes the machine · the same call twice does no more than once

Set a machine's G54 work zero on its blank - the front-left top corner (default: the corner at the machine's front left, X and Y least, when the blank lies square - turned over "v" too) or the top centre, with the tool now in the spindle as the length reference; an optional offset [x, y, z] mm. Homes first if needed.

Parameters

Name	Type	Default	Description
`corner`	string	`"front-left-top"`	(one of `front-left-top`, `center-top`)
`offset`	array of number	`[0, 0, 0]`	
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `g54_mpos`.

`inspect_part` - Inspect the part with the probe

changes the machine

Check the finished part on the machine with the touch probe in the spindle against its design: from the features of a program program_generate made (program; or features as it takes them) the checks - a pocket's sizes, centre and depth, a hole's diameter and position, a slot's width and depth, a faced step, a rectangle's or a circle's contour - each measured by a probe cycle (as probe_cycle runs them: results from the trigger points) and judged against the design (the tolerance given, else ISO 2768-1 - class f unless tol.class says) and against the twin's prediction (the same touches on the workpiece the twin cut: the twin's estimate, not a measuring uncertainty). Depths from a touch on the top where no feature reaches. A chamfer: two touches down onto its face - its angle, its width from the outline and its depth there (its V-bit: the feature's tool, else the job's; the material's side from the job's pocket or contour with that outline). V-carving, engraving, threads and 3D surfaces are said not measured. ISO 2768-2's class after part 1's as a drawing writes it (tol.class "mK") adds form checks and touches: a floor's flatness (a grid of 9 instead of 1), the walls' perpendicularity to Z (4 more at a second depth), a long wall's straightness (5 along it, 20 mm and longer), a circle's circularity (8 round instead of 4) and its axis' perpendicularity (4 more) - the answer's form_cost and the window's question say how many and how long before; only: [..] does part of them; one letter adds none. Least-squares estimates from a few touches, not ISO 1101's minimum zone nor a CMM's form measurement (summary.form_method). The work offset must be the one the part was cut with. Each inspection is kept as a record (written once, its SHA-256 in the audit log): inspect_report makes its CSV and PDF. plan: true only says what it would do. On the window's machine the touches take their machine time at the simulation speed the user set; longer than one call may wait, the call says how to measure it in calls that fit (only: [..] each) and the speed that fits one; continue: "last" makes each batch after the first go on from the one before - one record, one report. Professional edition.

Parameters

Name	Type	Default	Description
`program`	string		a program program_generate made: the features of the job kept with it
`version`	integer		that job's version (0: the last)
`features`	array of object		instead of program: the features as program_generate takes them
`tool`	string		with features: the tool they were cut with (a drilling's holes are its size)
`only`	array of integer		only these features (1, 2 ... in the job)
`continue`	string		a batch: the inspection it goes on from ("last", or its record) - the same program, set-up and work offset; the record and its report then hold all the batches (a feature measured again: this batch's)
`tol`	object		+/- mm (an angle: degrees) instead of ISO 2768-1
`tol.size`	number		

Name	Type	Default	Description
`tol.depth`	number		
`tol.position`	number		
`tol.angle`	number		a chamfer's, degrees
`tol.class`	string	`"f"`	ISO 2768-1's class where no tolerance is given (a drawing's ISO 2768-mK: m) - with ISO 2768-2's after it (mK: H, K or L) its form checks too, more touches (one of `f`, `m`, `c`, `v`, `fH`, `fK`, `fL`, `mH`, `mK`, `mL`, `cH`, `cK`, `cL`, `vH`, `vK`, `vL`)
`probe`	object		the probe's compensation instead of this machine's calibration: r_eff, lobing, phase_deg, pretravel_z
`plan`	boolean	`false`	only plan it, nothing moves and nothing is recorded: each feature, what it measures there or why not, the touches and about how long, the form checks' share, the classes, the program zero it would measure in (zero_mpos), each touch's point and way (points) - with the spindle's probe, else the tool table's first, else probe 2 L30 (planned_with)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `plan`, `features`, `summary`, `reference`, `cycles`, `twin_cell_mm`, `inspection`, `record`, `sha256`, `record_error`, `error`, `form_cost`.

`inspect_report` - Inspection report

changes the machine · the same call twice does no more than once

The report of an inspection inspect_part recorded, written by the program itself: a CSV (UTF-8, for a spreadsheet) and a PDF - the job, setup, tools, the probe and its ring-gauge calibration, each feature's nominal, tolerance and its basis (ISO 2768-1's class and table, or given), the twin's estimate, the measured value, the deviation and the verdict, why a feature was not measured, and the twin's workpiece from above. Into the reports folder (Documents/Twinwright Studio/Reports); their SHA-256 into the audit log. The same record gives the same bytes again. The twin's estimate is the twin's own calculation for the same touches on its simulated workpiece, not the machine's measuring uncertainty; the report says nothing about how precisely the machine measures or positions. The PDF is in English where the language in use needs letters its fonts lack (the CSV is in it). verify: a file checked instead - its SHA-256 searched for in the audit log and the log's chain checked. It moves nothing. Professional edition.

Parameters

Name	Type	Default	Description
`inspection`	string	`"last"`	the record's name as inspect_part gave it (last: the one recorded last)
`format`	string	`"both"`	(one of `csv`, `pdf`, `both`)
`unit`	string		lengths (default: the units in use) (one of `mm`, `in`)
`paper`	string		the PDF's (default: the region's) (one of `a4`, `letter`)
`verify`	string		a file's path: check it against the audit log instead of writing a report

Returns (structured content): `machine`, `notice`, `inspection`, `record`, `record_sha256`, `csv`, `pdf`, `found`, `chain_whole`, `sha256`.

`setup_sheet` - Setup sheet

changes the machine · the same call twice does no more than once

The setup sheet of a job program_generate made, for the operator at the machine: one HTML page to print (A4 or Letter: the region's) in the language in use - each program's name, lines and SHA-256 (to match it with the one loaded), the blank and its fixture as the twin sets it up (the vise's parallels, the grip), the tools, each setup's way there (turned over, held another way), its work zero and the probe panel's cycles that find it, the program's travel, its Z clearance (the probe and each tool over the set-up and to the program's highest and lowest points - the twin's Z budget: enough, or how much short), the operations (strategy, tool, speed, feed, depth, the twin's estimate of each one's time), a top view, the notes, a box to sign. The blank as the machine has it set up (the call's machine), else the job's. Into the reports folder (Documents/Twinwright Studio/Reports) as <job>-setup-sheet.html; its SHA-256 into the audit log. The times are the twin's estimates from the program, not measured. It moves nothing. Professional edition.

Parameters

Name	Type	Default	Description
`program` *	string		the name program_generate was given (its job is kept with it; a job in setups: each setup's program)
`version`	integer		the job's version (left out: the last)
`paper`	string		left out: the region's (one of `a4`, `letter`)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `file`, `sha256`, `bytes`, `paper`, `programs`, `z_clearance`.

`probe_cycle` - Touch-probe cycle

changes the machine

Measure with the touch probe in the spindle (setup_tool "probe 2 L30") and, if asked, set the work zero from what it finds: cycle edge (a face of the blank: side x-, x+, y-, y+), corner (front-left, front-right, back-left, back-right; inside: a pocket's corner), bore (its centre and diameter, 4 or 8 points), boss (the same from outside), surface (its Z) or calibrate (a ring gauge of known d: the probe's effective ball radius, its lobes and pretravel - this machine's later cycles here use them). Each touch: a protected move to its start (G38.3), a fast touch, the release (G38.4), a slow touch (F25 at most); the results come from the slow touches' trigger points ([PRB:]), not from where the axes stopped. Left out, at comes from the blank the twin knows. Refused: no probe in the spindle, its input already triggered, the spindle turning, a move outside the soft limits. set {x, y, z}: the active work offset moved so the feature gets these work coordinates (G10 L2); the result's set_parts give each axis found as its G10 L2 word (the feature at work 0, or at set's value). Uncalibrated, the results are off by the probe's pretravel. dry: the plan only - checks (probe, machine, soft limits, safe height, depth: ok, warn or bad), each touch's path (machine coordinates), the program and its time; nothing moves. calibrate keep false: its numbers and a cal_id, kept later with save (this session's measurement only).

Parameters

Name	Type	Default	Description
`cycle` *	string		(one of `edge`, `corner`, `bore`, `boss`, `surface`, `calibrate`)
`side`	string		edge: the face (x- the blank's least X, touched from outside moving +X; y- its front) (one of `x-`, `x+`, `y-`, `y+`)
`corner`	string	`"front-left"`	(one of `front-left`, `front-right`, `back-left`, `back-right`)
`inside`	boolean	`false`	corner: a pocket's corner, its walls touched from inside (at needed)
`at`	array of number		[x, y] in the active work coordinates: a point on the face (edge), the corner, the centre (bore, boss, calibrate), where to touch (surface) - left out: from the blank
`d`	number		bore, boss: the nominal diameter; calibrate: the ring gauge's known diameter, mm (left out: a tube's or cylinder blank's)
`depth`	number	`3`	how far below the top the ball's centre touches, mm
`top`	number		the work Z the depth is from (left out: the blank's top)
`points`	integer		edge 1, or 2 (span apart: the face's turn too), 3, 4 or 8 over span (its straightness too: about their least-squares line); corner 2, or 3 (two on its X face: the blank's turn and where its faces meet); bore, boss 4 (default) or 8 (its roundness too) (one of `1`, `2`, `3`, `4`, `8`)
`span`	number		mm between the first and the last touch on one face (left out: 0.6 of the face)
`grid`	array of integer		surface: [nx, ny] touches over size (25 at most): the plane through them - its tilt and flatness
`size`	array of number		surface grid: [w, h] mm about at (left out: 0.6 of the blank)
`inset`	number	`5`	corner: mm from the corner along each face
`clear`	number	`3`	mm between the probe and where the face should be when a touch starts
`search`	number	`5`	mm past where it should be a touch looks
`fast`	number	`300`	
`slow`	number	`20`	(at most 25)
`set`	object		move the active work offset so the feature found gets these work coordinates (the axes the cycle finds)
`set.x`	number		
`set.y`	number		
`set.z`	number		
`probe`	object		the probe's compensation instead of this machine's last calibration here: r_eff, lobing, phase_deg, pretravel_z (mm, degrees)
`dry`	boolean	`false`	the plan only: the checks, the path, the program and its time - nothing moves, nobody is asked
`keep`	boolean	`true`	calibrate: keep the calibration on the machine (false: its numbers and a cal_id only)
`save`	string		calibrate: keep the calibration this session measured with keep false (its cal_id) - nothing measured
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `cycle`, `touches`, `result`, `work`, `set`, `comp`, `archive`, `notes`, `program`, `sent`, `error`, `dry`, `checks`, `plan`, `seconds`, `set_parts`, `cal_id`.

`setup_part` - Set the design part

changes the machine · the same call twice does no more than once

The design part (an STL of the finished part, mm) to compare a machine's workpiece with. origin: machine mm of the part's origin; omitted: the program zero (as CAM writes it). An empty stl takes it away.

Parameters

Name	Type	Default	Description
`stl` *	string		
`origin`	array of number		
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `part`.

`setup_copy` - Copy a setup

changes the machine (destructive) · the same call twice does no more than once

Copy one machine's setup - blank and fixture, the tool in the spindle, the tool table, G54, the design part - onto an invisible practice machine, to try programs there at full speed (default: from the user's window to "sandbox"). The workpiece comes as it is there, cuts and all (stock: "fresh" for a new blank).

Parameters

Name	Type	Default	Description
`stock`	string	`"current"`	current: the machine's workpiece as it is, cuts and all (carry on with a part already cut); fresh: a new blank (a whole program from the start) (one of `current`, `fresh`)
`from`	string		the machine to copy (default: the user's window)
`to`	string	`"sandbox"`	a practice machine: sandbox or sandbox-<name>

Returns (structured content): `machine`, `notice`, `copied_from`, `workpiece`, `tool`.

`setup_example` - Example parts

changes the machine (destructive)

Without id: the example parts that come with the twin (id, title, material, blank, summary). With id: a machine set up as the example's operator would - blank, tool table, homing, G54 on the blank, the design part - with the example's program loaded, not started. To verify an example without setting anything up: program_simulate {example: id}.

Parameters

Name	Type	Default	Description
`id`	string		
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `examples`, `example`, `ready`.

Programs

`program_check` - Check a program

read-only · the same call twice does no more than once

Check a program without running it: the controller's own check mode ($C) - every line it refuses, with the error explained - the preview of its path against a machine's setup (estimated time, collisions, rapids into the stock, travel) and the static rules S01-S22 (start block, end, F and S missing or out of range, dense short lines, plunges, rapids below Z0, the water-cooled spindle without M8, arcs in inverse time, G33 without a spindle encoder, characters the controller takes for real-time commands, drilling cycles' holes a single-precision controller plunges once more, Z down after a tool change before X and Y, a work coordinate system turned about Z - the twin does not turn it yet, the Z surface map ($HM) on - the twin does not follow its offsets yet, ...) with a suggested line for each. Runs on a hidden copy of the machine's setup, its workpiece as it is (stock): the machine does not change.

Parameters

Name	Type	Default	Description
`gcode`	string		the program's text - or leave it out and give name (and version) of a program in the working folder (program_versions)
`name`	string	`"program.nc"`	
`version`	integer		with name, no gcode: that saved version's text (0: the last)

Name	Type	Default	Description
`file`	string		instead of gcode: the program's path on this computer (a .nc .ngc .gcode .tap ... file, up to 32 MB) - read and kept in the working folder as a version under its name
`stock`	string	`"current"`	current: the machine's workpiece as it is, cuts and all (carry on with a part already cut); fresh: a new blank (a whole program from the start) (one of `current`, `fresh`)
`example`	string		an example's setup (and program, when gcode is omitted) instead of the machine's: see setup_example
`setup`	object		an explicit setup instead of the machine's: {spec: the workpiece's setup as machine_status shows it (status.workpiece), tool, tools: {T: spec}, g54: [x,y,z] MPos, part: STL path}
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `ok`, `refused`, `preview`, `static_rules`.

`program_simulate` - Simulate a program

read-only · the same call twice does no more than once

Run a program to its end on a hidden copy of a machine's setup - the machine itself does not change; the copy starts from the machine's workpiece as it is, cuts and all (stock: "fresh" for a new blank) - as fast as the computer goes: moves hundreds of times faster than real time, cutting a few times faster (a heavy cut can be slower than real time). The report: completed or not, machine time, alarms, collisions, near misses, rapids into material, cuts with the spindle stopped, workpiece slip, chatter margin, force and power peaks, removed volume, deviation from the design part, the slowest lines - and a run id for run_profile, result_compare and result_image. Programs with M6 need tools. Longer than budget_s: the answer is the run id (run_status waits).

Parameters

Name	Type	Default	Description
`gcode`	string		the program's text - or leave it out and give name (and version) of a program in the working folder (program_versions)
`name`	string	`"program.nc"`	
`version`	integer		with name, no gcode: that saved version's text (0: the last)
`file`	string		instead of gcode: the program's path on this computer (a .nc .ngc .gcode .tap ... file, up to 32 MB) - read and kept in the working folder as a version under its name
`tools`	object of string		the tool table for M6: {"1": "flat 6 n3 L20", ...}
`stock`	string	`"current"`	current: the machine's workpiece as it is, cuts and all (carry on with a part already cut); fresh: a new blank (a whole program from the start) (one of `current`, `fresh`)
`example`	string		an example's setup (and program, when gcode is omitted) instead of the machine's: see setup_example
`setup`	object		an explicit setup instead of the machine's: {spec: the workpiece's setup as machine_status shows it (status.workpiece), tool, tools: {T: spec}, g54: [x,y,z] MPos, part: STL path}
`check_first`	boolean	`true`	
`timeout_s`	number	`3600`	simulated seconds allowed
`budget_s`	number	`40`	wall-clock seconds this call waits before it answers with a run id (at most 45: clients give up after 60 s; run_status waits for the rest)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `run`, `completed`, `simulated_time_s`, `issues`, `status`.

`program_run` - Run a program on the machine

changes the machine (destructive)

Run a program on the machine itself - the one loaded, or gcode (it replaces the loaded one): on the user's machine in real time, after the user allows it in the window; on a practice machine (sandbox) at full speed, cutting its workpiece. Checked first, the loaded program too (check_first): not run when the controller refuses a line or the check ends in an alarm - and on the user's machine not when the check's preview finds a collision, a rapid move into material, a cut with the spindle stopped or a move out of travel. Same report as program_simulate. To find out first what a program would do, use program_simulate.

Parameters

Name	Type	Default	Description
`gcode`	string		the program's text - or leave it out and give name (and version) of a program in the working folder (program_versions)
`name`	string	`"program.nc"`	

Name	Type	Default	Description
`version`	integer		with name, no gcode: that saved version's text (0: the last)
`file`	string		instead of gcode: the program's path on this computer (a .nc .ngc .gcode .tap ... file, up to 32 MB) - read and kept in the working folder as a version under its name
`tools`	object of string		the tool table for M6: {"1": "flat 6 n3 L20", ...}
`check_first`	boolean	`true`	
`fits_only`	boolean	`false`	nothing run, nothing moved: only the twin's grid made finer where the touch probe will measure the fits of the saved program name (its CAM job), as a run does before it starts - for a program started some other way (the window's own start); the answer's fit_regions says how many regions, where (program mm) and their cells, or why none (no job, no fits, several setups). Before the cut, never after: what is cut already stays as coarse as it was
`timeout_s`	number	`3600`	simulated seconds allowed
`budget_s`	number	`40`	wall-clock seconds this call waits before it answers with a run id (at most 45: clients give up after 60 s; run_status waits for the rest)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `run`, `completed`, `simulated_time_s`, `issues`, `status`.

`program_load` - Load a program

changes the machine · the same call twice does no more than once

Load a program into a machine without starting it. On the user's machine the window shows how it differs from the program loaded there and the user allows it; they may then start it themselves.

Parameters

Name	Type	Default	Description
`gcode`	string		the program's text - or leave it out and give name (and version) of a program in the working folder (program_versions)
`name`	string	`"program.nc"`	
`version`	integer		with name, no gcode: that saved version's text (0: the last)
`file`	string		instead of gcode: the program's path on this computer (a .nc .ngc .gcode .tap ... file, up to 32 MB) - read and kept in the working folder as a version under its name
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `lines`.

`program_debug` - Debug a program

read-only

Debug a program on a hidden copy of a machine's setup. run_to: runs gcode from its start until a file line (line) or a condition (when: force_n, load_pct, ferr_mm, margin, near_mm as limits; cut, rapid_cut, collision, alarm, state true; modal ["units","wcs","tool","spindle",...]) and stops virtual time there - mid-motion, exactly as the program has it, at the same instant every time. Then inspect (the line, modes, positions, speed, the cut, the planner's queue), step (until: time with ms, block, or pause), continue (to the next pause or the end), end.

Parameters

Name	Type	Default	Description
`action` *	string		(one of `run_to`, `inspect`, `step`, `continue`, `end`)
`gcode`	string		the program's text
`name`	string	`"program.nc"`	
`file`	string		instead of gcode: the program's path on this computer (a .nc .ngc .gcode .tap ... file, up to 32 MB) - read and kept in the working folder as a version under its name
`line`	integer		file line (1-based)
`when`	object		conditions, e.g. {"force_n": 60, "cut": true}
`tools`	object of string		the tool table for M6: {"1": "flat 6 n3 L20", ...}
`until`	string	`"time"`	(one of `time`, `block`, `pause`)

Name	Type	Default	Description
`ms`	number	`10`	
`stock`	string	`"current"`	current: the machine's workpiece as it is, cuts and all (carry on with a part already cut); fresh: a new blank (a whole program from the start) (one of `current`, `fresh`)
`example`	string		an example's setup (and program, when gcode is omitted) instead of the machine's: see setup_example
`setup`	object		an explicit setup instead of the machine's: {spec: the workpiece's setup as machine_status shows it (status.workpiece), tool, tools: {T: spec}, g54: [x,y,z] MPos, part: STL path}
`timeout_s`	number	`3600`	simulated seconds allowed
`budget_s`	number	`40`	wall-clock seconds this call waits before it answers with a run id (at most 45: clients give up after 60 s; run_status waits for the rest)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `reached`, `pause`, `where`.

`program_restart` - Restart from a line

changes the machine (destructive)

The program loaded on a machine from a file line on, as an operator restarts after a broken tool: a start block from the program's own state before that line (units, plane, work offset, tool, spindle speed and direction with its spin-up, coolant, over the start point at the safe height, down to it, feed, distance mode), then the rest. start: true loads and runs it (on the user's machine after the user allows it).

Parameters

Name	Type	Default	Description
`line` *	integer		
`start`	boolean	`false`	
`safe_z`	number		machine Z to go up to first (default 2 mm below home)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `line`, `started`, `program`.

`program_optimize` - Optimize a program

changes the machine · the same call twice does no more than once

Make a program faster without asking more of any tool than the program already does: it runs on a hidden copy of the setup; lighter cuts get feeds up to the heaviest cut the same tool already makes (or the limits in goals), moves in air go at the axes' top speed, finishing passes and plunges keep theirs, a chattering tool gets the steadiest speed near its own (0.7 to 1.8 times it, the depth within 10 % of it the deepest) where that takes the cut half as deep again, collinear lines are merged, rapid retracts come down to just over the blank and the fixture, arcs are fitted where the sender could not keep up. The result runs again on a second copy (lines over a limit are lowered and it runs once more). Answers with the program, its changes and the comparison; saved as a version (author optimizer) when it is better in every respect. Takes two or three runs of the program: a run id comes back after budget_s.

Parameters

Name	Type	Default	Description
`gcode`	string		the program's text
`name`	string	`"program.nc"`	
`file`	string		instead of gcode: the program's path on this computer (a .nc .ngc .gcode .tap ... file, up to 32 MB) - read and kept in the working folder as a version under its name
`stock`	string	`"current"`	current: the machine's workpiece as it is, cuts and all (carry on with a part already cut); fresh: a new blank (a whole program from the start) (one of `current`, `fresh`)
`example`	string		an example's setup (and program, when gcode is omitted) instead of the machine's: see setup_example
`setup`	object		an explicit setup instead of the machine's: {spec: the workpiece's setup as machine_status shows it (status.workpiece), tool, tools: {T: spec}, g54: [x,y,z] MPos, part: STL path}
`goals`	object		
`goals.feeds`	boolean	`true`	
`goals.speeds`	boolean	`true`	
`goals.merge`	boolean	`true`	

Name	Type	Default	Description
`goals.retracts`	boolean	`true`	
`goals.arcs`	boolean		default: when the sender could not keep up
`goals.chip_mm`	number		
`goals.force_n`	number		
`goals.load_pct`	number		
`goals.load_cap`	number	`80`	
`goals.feed_max`	number		
`goals.arc_acc`	number		mm/s2 towards an arc's centre: its feed no more than 60 sqrt(arc_acc r) - the controller's (the MC-1's kernel: sqrt(3)/2 of the axes' 300 = 260); default 0, no bound
`goals.air_feed`	number		
`goals.finish_ae`	number	`0.15`	
`timeout_s`	number	`3600`	simulated seconds allowed
`budget_s`	number	`40`	wall-clock seconds this call waits before it answers with a run id (at most 45: clients give up after 60 s; run_status waits for the rest)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `optimized`, `verdict`, `gcode`, `status`, `run`.

`geometry_import` - Import a shape

changes the machine · the same call twice does no more than once

Make a shape the programs can cut, from what you write - SVG path data (svg_path, its numbers in mm: e.g. a heart "M 10,30 A 20,20 0,0,1 50,30 A 20,20 0,0,1 90,30 Q 90,60 50,90 Q 10,60 10,30 z"), an SVG or DXF drawing (svg, dxf: the text) or file (an .svg or .dxf path; a DXF's layers: [names] reads only those), formulas in t (curve: {x: "16*sin(t)^3", y: "13*cos(t)-5*cos(2*t)-2*cos(3*t)-cos(4*t)", t: [0, "2*pi"]}), basic shapes added or cut in order (shapes: rect, circle, ellipse, slot, polygon, star, path), single-stroke text (text, font, text_height, align) or points - instead of working out its points yourself. Then where it goes: width_mm or height_mm (scaled to fit), scale, rotate, mirror, and center [x, y] (its box's middle) or at [x, y] (its lower left corner) in the program's mm (Z0 on the blank's top). SVG's y points down: it is turned the right way up. Saved under name as a version (the same shape again: not saved again); the answer says its size and box, outlines and islands, area, the sharp inner corners a pocket's tool cannot get into, the largest tool that fits and the narrowest place, its symmetry - with a picture: look at it before cutting. Then program_generate {features: [{op: "pocket", geometry: name, depth}]} (or contour, engrave). A mesh (file: .stl, .obj, .3mf, .glb, .gltf; units, up, width_mm or scale, depth_mm, rotate, mirror, center or at, top) is saved the same way for 3D: rough3d and finish3d take it. With only name: a saved shape again (version: an older one); with nothing: the shapes there are. Units are mm; a shape is data - it does not touch a machine.

Parameters

Name	Type	Default	Description
`name`	string		the shape's name: saved under it, used by it (geometry: name)
`version`	integer		with only name: that saved version (default the last)
`svg_path`	string		SVG path data: M L H V C S Q T A Z, lower case relative
`svg`	string		an SVG document's text
`dxf`	string		a DXF drawing's text (ASCII): lines, arcs, circles, polylines, splines, ellipses, blocks
`file`	string		an .svg or .dxf file's path - or a mesh for 3D: .stl, .obj, .3mf, .glb, .gltf
`layers`	array or string		DXF: only these layers (default: all that are on)
`join`	number	`0.01`	DXF: mm within which ends are joined into closed outlines
`curve`	object		formulas in t (radians; sin cos tan sqrt abs exp ln pow pi ...)
`curve.x` *	string		
`curve.y` *	string		
`curve.t`	array of ['number', 'string']		
`curve.closed`	boolean	`true`	
`shapes`	array of object		rect {x,y,w,h[,r]}, circle {cx,cy,d}, ellipse {cx,cy,rx,ry}, slot {x0,y0,x1,y1,w}, polygon {points} or {cx,cy,n,d}, star {cx,cy,n,d,d_inner}, path {svg}; op: add, cut, intersect
`text`	string		text: single strokes to engrave (font simplex, duplex, script: Latin letters, digits, signs) or the letters' outlines to pocket or V-carve (an outline font)

Name	Type	Default	Description
`font`	string		simplex (default), duplex, script: single strokes; sans (Noto Sans, comes with Studio) or a font of the computer's by its name - "微软雅黑", "宋体", "Arial" ... (Chinese: a computer's font): outlines
`text_height`	number		the capitals' height, mm (10)
`align`	string		(one of `left`, `center`, `right`)
`points`	array of array		a polygon: [[x, y], ...]
`units`	string	`"mm"`	svg_path's numbers; a DXF's that does not say its own; an STL's or OBJ's (mm, cm, m, in) (one of `mm`, `cm`, `m`, `in`, `pt`, `px`)
`up`	string	`"z"`	a mesh: which of its file's axes points up (most OBJ: y) (one of `z`, `-z`, `y`, `-y`, `x`, `-x`)
`depth_mm`	number		a mesh: its Z size (Z alone: a relief made shallower)
`top`	number	`0`	a mesh: the Z its highest point goes to
`bottom`	number		a mesh: the Z its lowest point goes to (instead of top)
`px_per_in`	number	`96`	
`flip_y`	boolean		
`scale`	number		
`width_mm`	number		
`height_mm`	number		
`mirror`	string		(one of `x`, `y`)
`rotate`	number		degrees, counter-clockwise
`center`	array of number		
`at`	array of number		
`tolerance`	number	`0.005`	mm: curves that are not circles become lines this close
`fill_rule`	string	`"evenodd"`	(one of `evenodd`, `nonzero`)

Returns (structured content): `machine`, `notice`, `name`, `version`, `size_mm`, `bbox`, `shapes`.

`geometry_nest` - Lay parts out on the blank

read-only · the same call twice does no more than once

Lay several parts out on the machine's blank (or a plate given) and cut them out - nesting. Each part a shape: geometry (a name from geometry_import) or one given as geometry_import takes it (svg_path, dxf with layers, shapes ...); count copies of it, each turned by quarter turns where that fits them better (turns: 0, 90, 180, 270; default all), its own features (holes, pockets ...) written in its shape's coordinates. They go in the largest first, each where it fits lowest, then leftmost - the shapes, not their boxes: a small part in a larger one's notch or its hole (in_holes) - gap apart (default the tool's diameter + 1 mm) and margin (1) inside the plate (default the blank's top). Nothing is machined: the answer says where each copy went, how much of the plate the parts use, the copies that did not fit (placed less than asked), with a picture; and features - each copy's own where it went, then every copy cut out, those in holes first (a contour outside its shape, depth deep - default through the blank and 0.2 - with tabs, 3 unless given) - for program_generate as they are, with nest (kept with the job).

Parameters

Name	Type	Default	Description
`parts` *	array of object		the parts, 1 to 16, e.g. [{geometry: "bracket", count: 4}, {shapes: [{type: "circle", cx: 0, cy: 0, d: 20}], count: 3}]
`plate`	array of number		[x0, y0, x1, y1] where the parts go, the program's mm (default the blank's top)
`depth`	number		how deep each part is cut out (mm; default the blank's thickness + 0.2)
`gap`	number		mm between two parts (default the tool's diameter + 1)
`margin`	number	`1`	mm from the plate's edge to the parts' outlines
`tabs`	integer or object		the cut-outs' tabs: a count or {count, width, height, holes: false - none round the parts' holes} (default 3)
`tool`	string		the tool the gap is for (default the one in the spindle)
`setup`	object		the setup to measure the blank and the tool on instead of the machine's, as program_generate takes it (spec, tool, tools, g54): a hidden copy set up so
`in_holes`	boolean	`true`	a small part may go into a larger one's hole (cut out before it); false: the holes taken as filled
`outlines`	boolean	`false`	each placed copy's outline where it went and each part's own: {outer: [[x, y], ...], holes: [...]}; each copy's tabs where the program will leave them: [[x, y, nx, ny], ...] on its outline, the normal away from its material - for drawing the layout

Name	Type	Default	Description
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `placed`, `parts`, `used_pct`, `features`, `nest`.

`program_generate` - Generate a program from features

changes the machine · the same call twice does no more than once

Write a program from machining features instead of line by line: face, pocket, slot, contour (outside, inside or on an outline; tabs hold a part cut through), drill, bore (helix), ream (a reamer widening holes drilled before to its size and class: drill, then ream), engrave (single-stroke text, or a shape's lines), chamfer (a V-bit along edges), thread (a thread mill in a bored hole), vcarve (a V-bit's tip along an outline's middle lines, each point as deep as the V reaches: lettering in an outline font, any closed shape). A pocket or a contour takes any outline, islands and all: a shape made with geometry_import (geometry: its name), SVG path data right in the feature (svg_path with geometry_import's width_mm, center ...), a rectangle with corner radius, a circle, polygon points, or the closed outline another program cuts (from_program). E.g. {op: "pocket", geometry: "heart", depth: 4}; {op: "pocket", svg_path: "M 10,30 A 20,20 0,0,1 50,30 A 20,20 0,0,1 90,30 Q 90,60 50,90 Q 10,60 10,30 z", width_mm: 40, center: [30, 30], depth: 4}; {op: "pocket", from_program: "heart.nc", offset: 3, depth: 13} (a hollow where heart.nc cut the heart's outline; offset: grown by the tool's radius). A pocket is cleared in rings from its deepest spot out - or, strategy "adaptive", by a walk that cuts a set width (ae, 10 % of the tool) all the way: deeper layers, a faster feed for the thinner chips, no slotting but the entry; or "zigzag", straight passes to and fro - its walls, round the outside and every island, finished to size (a slot: strategy "trochoidal", loops along it); the answer says where the tool cannot reach (corners narrower than its radius stay round). Speeds, feeds and depths come from the blank's material and the tool, within the spindle's range: with verify, the twin's stability model picks the speed and keeps each depth (roughing steps, the walls' finishing passes) to 80 % of the stable one before the program is written. Ramps and helices into material, climb milling. The clearance height stays within what the setup leaves above the blank. With verify (default) it runs on a hidden copy of the setup, its workpiece as it is (stock), and where the tool would still chatter it is written again shallower there - a program too long to try within the call likewise behind it (verification.check_run: run_status gives the last version's report, closed_loop what it changed). Saved as a version (author generator; each one the loop tried too). operations says each feature's lines, layers and values - the first 64; past them (a nest's) feature_spans gives every feature's [first line, last line, layers, tool]. A hole's or a shaft's diameter (a circle's pocket or contour, a bore, a reaming) may carry its fit (ISO 286): d "Ø8 H7", or d 8 with fit "H7", a pair "H7/g6" (the hole takes H7, the shaft g6) or a common fit (sliding, locating, transition, press), or tol "+0.015/0" - cut to the middle of its zone, a spring pass where the twin's estimate of the tool's give asks for one (operations[].fit says what and why: the twin's estimate). So may a slot's width (width "6 N9", or width 6 with fit "N9": a keyway's N9, P9, JS9 - cut with a cutter narrower than it, its walls to size) and a rectangle's pocket or contour, both its sizes (fit "H7"; outside: a shaft's class, a key's h9). The program's zero: the machine's G54 (setup_work_zero). A part machined from two sides or more: setups instead of features - a program each, setup 1 on the machine as it is, each later one turned over (turn) or held another way (fixture) from the one before, G54 set again on the blank (zero); the job and every program saved together.

Parameters

Name	Type	Default	Description
`stock`	string	`"current"`	current: the machine's workpiece as it is, cuts and all (carry on with a part already cut); fresh: a new blank (a whole program from the start) (one of `current`, `fresh`)
`features`	array of object		each {op, ...}: face {x0,y0,x1,y1,depth}; pocket {geometry name | svg_path (+ width_mm, center) | shape rect x0,y0,x1,y1[,r] | shape circle cx,cy,d | shape polygon points [[x,y],..] | from_program name[,version]; [offset]; [strategy adaptive[, ae] | zigzag[, angle]]; [rest true]; depth}; slot {x0,y0,x1,y1[,width],depth[,strategy trochoidal[,step]]}; contour {the same outlines, side outside|inside|on, depth}; drill {points [[x,y]..],depth[,peck]}; bore {points,d,depth}; engrave {text,x,y,height,depth[,font] | geometry name, depth[, surface: a mesh's name - on a curved top, depth under it]}; chamfer {an outline, width} (a V-bit); thread {points,d,pitch,depth[,hand]} (a thread mill); groove {points,d,bore,depth[,width,step]} (a T-slot cutter: a ring cut into a round hole's wall); perlage {an outline, depth[, pitch, stagger, margin, dwell]} (a flat tool: overlapping spots over a face); vcarve {an outline, or text with font sans or a computer's font, text_height, center[, max_depth]} (a V-bit); rough3d {geometry: a mesh's name[, allowance, stepdown, ae, area, rest_from, rest_allowance, rest_stepdown]} (a flat end mill); finish3d {geometry[, strategy, scallop or stepover, angle, steep_angle, area, rest_from]} (a ball end mill; strategy pencil: along the creases); contour tabs: n or {count,width,height,holes}
`setups`	array of object		instead of features: a part machined from two sides or more, a program each, in order. Setup 1 is the machine's setup as it is; each later one turned over (turn) or held another way (fixture) from the one before, G54 set again on the blank (zero). Within the call each later program is written for the blank as it was (what the setups before cut is not known yet). name, tool, material, cutting and safe_z here are every setup's defaults. E.g. setups: [{features: [{op: "pocket", shape: "rect", x0: 8, y0: 8, x1: 24, y1: 22, depth: 3}]}, {turn: "u", features: [{op: "face", x0: 0, y0: 0, x1: 40, y1: 30, depth: 1}]}]
`nest`	object		geometry_nest's nest (its parts laid out on a plate): kept with the job beside the features it made; given without features, the parts are laid out here (plate, depth and the parts needed) - not with setups
`tool`	string		default: the machine's tool in the spindle
`material`	string		default: the machine's blank's; a code of the machine's table (6061-T6, 45, 304 ...) or another standard's name for one (EN AW-6061, C45, SUS304 ...: read as that code, the nearest grade - for its cutting data)
`cutting`	object		rpm, feed, ap (depth of cut), ae (stepover), finish_ap to use instead
`setup`	object		the setup to generate for and verify on instead of the machine's (as program_optimize takes it)
`safe_z`	number	`5`	the clearance height over Z0; kept within what the setup leaves above the blank
`name`	string	`"generated.nc"`	
`title`	string		the job's name to show (one line, 80 characters at most; kept with the job, not in the program)
`description`	string		a line about the job (200 characters at most; kept with the job)

Name	Type	Default	Description
`verify`	boolean	`true`	
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `gcode`, `verification`, `operations`, `setups`, `total`.

`program_edit` - Change a program by intent

changes the machine · the same call twice does no more than once

Change a program without sending it again line by line: edits applied in order - set_s value (every S), set_f value or scale (every F), shift_z by and scale_z factor (the cuts: Z at or below the program's zero; all: every Z), split into (from..to: one pass cut in that many layers, a ramp kept a ramp - back up and over to its start in between), safe_z z (the clearance height: every Z above zero); from/to: file lines (default all). G53, G28, G30, G10 and G92 lines are left alone; programs in absolute mm (G90 G21). The program: gcode, or name (and version: default the last) from the working folder. The result is saved as a new version (save false: not) with the difference; check it with program_check {name, version}.

Parameters

Name	Type	Default	Description
`gcode`	string		the program's text (or name and version)
`name`	string		
`version`	integer		with name, no gcode: that saved version (0: the last)
`edits` *	array of object		each {op, ...}: set_s {value}; set_f {value | scale}; shift_z {by[, all]}; scale_z {factor[, all]}; split {from, to, into}; safe_z {z}; any of them with from, to (file lines)
`save`	boolean	`true`	

Returns (structured content): `machine`, `notice`, `gcode`, `version`, `changed_lines`, `diff`.

`program_versions` - Program versions

changes the machine · the same call twice does no more than once

Program versions in the working folder (Documents/Twinwright Studio/programs/<name>/vNNNN.nc with who, when and why). action list: the versions (without name: every program, the last changed first); get: a version's text (default the last); save: gcode as a new version (reason: what changed and why; the same text is not saved twice); diff: how b differs from a - texts, or name with a_version and b_version - with their simulation results side by side when both ran on this host. The user's own files are never written.

Parameters

Name	Type	Default	Description
`action`	string	`"list"`	(one of `list`, `get`, `save`, `diff`)
`name`	string		
`version`	integer		
`gcode`	string		
`reason`	string		
`a`	string		
`b`	string		
`a_version`	integer		
`b_version`	integer		

Returns (structured content): `machine`, `notice`, `name`, `version`, `versions`, `gcode`, `programs`.

`job_get` - A program's CAM job

read-only · the same call twice does no more than once

The CAM job a program was generated from - by program_generate or in the window's CAM page, the user's or an AI's - as it was kept with a version (default the last): its operations (features, or setups), tool, tool table, material, cutting values, safe_z, title; a line on each operation (job_edit numbers them so), who made it and when, its versions. An example loaded as shipped: the example's own job. Changes nothing.

Parameters

Name	Type	Default	Description
`name` *	string		the program's name (program_versions lists them)
`version`	integer		the job's version (0: the last)

Returns (structured content): `machine`, `notice`, `name`, `version`, `program_version`, `author`, `versions`, `job`, `operations`.

`job_edit` - Change a program's CAM job

changes the machine · the same call twice does no more than once

Change the CAM job a program was generated from (job_get shows it) and generate it again - as program_generate does, with its answer: the program's next version, the job's beside it. Each edit in turn: {feature: k, set: {depth: 3, tool: 2}} changes operation k's values (null removes one); {feature: k, remove: true}; {feature: k, move: j} moves it to place j, the others in their order (with set: changed too); {insert: k, op: {op: \"face\", ...}} a new operation before k (k one past the last: at the end); {set: {safe_z: 8, material: \"7075-T6\"}} the job's own values (tool, tools, material, cutting, safe_z, title, description - not its operations or name). A job in setups: setup: s in each edit. Nothing is changed or generated when an edit cannot be made.

Parameters

Name	Type	Default	Description
`name` *	string		the program's name
`version`	integer		the job's version to start from (0: the last)
`edits` *	array of object		
`verify`	boolean	`true`	
`stock`	string	`"current"`	current: the machine's workpiece as it is, cuts and all (carry on with a part already cut); fresh: a new blank (a whole program from the start) (one of `current`, `fresh`)
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `gcode`, `verification`, `operations`, `setups`, `total`.

Runs and results

`run_status` - Runs

changes the machine

Without run: the runs kept on this host (the last 16) - program, machine, who started it, running or done, a summary. With run: waits up to wait_s for it (at most 45 s: clients give up after 60 s) and gives its full report - how it ended first (done, stopped by whom, an alarm, out of time) and at which line; stop: true ends it early. A call that is cancelled or times out never stops a run.

Parameters

Name	Type	Default	Description
`run`	string		a run id (program_simulate's, program_run's), or "latest": your newest run
`wait_s`	number	`40`	at most 45
`stop`	boolean	`false`	

Returns (structured content): `machine`, `notice`, `run`, `completed`, `simulated_time_s`, `issues`, `status`, `runs`.

`run_profile` - Where a run's time went

read-only · the same call twice does no more than once

A run line by line (a run id from program_simulate or program_run; default the latest): the time each line took (they add up to the run), the feed asked for and reached and why it fell short (the axes' top speed, corners, arcs, the planner's look-ahead, a sender that could not keep up, the override, a hold), the cutting force, chip, chatter margin, spindle load, material removed, cutting air, events. sort: time (slowest first), risk (events first) or line; from/to: a range of file lines; paged (offset, limit).

Parameters

Name	Type	Default	Description
`run`	string		a run id (program_simulate's, program_run's), or "latest": your newest run
`sort`	string	`"time"`	(one of `time`, `risk`, `line`)
`limit`	integer	`20`	

Name	Type	Default	Description
`offset`	integer	`0`	
`from`	integer		
`to`	integer		
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `run`, `total_s`, `lines`.

`result_compare` - Compare with the design part

read-only · the same call twice does no more than once

How a machined workpiece deviates from its design part: cells within tolerance, material left above the part, cut into it. run: the workpiece a run left; else the machine's workpiece now.

Parameters

Name	Type	Default	Description
`run`	string		a run id (program_simulate's, program_run's), or "latest": your newest run
`tol`	number	`0.02`	
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `comparison`, `in_tolerance_pct`.

`result_image` - Picture of a result

read-only · the same call twice does no more than once

A picture (PNG) and its key numbers in words - kind: workpiece = the workpiece in 3D (view iso, top, front, right; color material, height, line = the program line that cut there last, deviation; path: a run's toolpath over it); toolpath = the path coloured by feed, force, chip, margin, line, z or limit (why the programmed feed was not reached), rapids dashed, the highest value marked with its line (view top or iso); deviation = the design part's deviation map from above, the worst places marked with their program lines (needs a part); profile = the slowest lines, coloured by what limited them; timeline = cutting force, spindle load, feed, chatter margin and the program line over time; comparison = two runs side by side (runs). run: a run of program_simulate - its end state; without it, workpiece and deviation show the machine now and the others its latest simulation (a simulation runs on a copy: to see its result, name its run). A picture larger than about 100 KB is drawn smaller (clients limit a result's size). The key numbers come in the text too.

Parameters

Name	Type	Default	Description
`kind`	string	`"workpiece"`	(one of `workpiece`, `toolpath`, `deviation`, `profile`, `timeline`, `comparison`)
`run`	string		a run id (program_simulate's, program_run's), or "latest": your newest run
`runs`	array of string		comparison: two run ids, A then B
`view`	string		(one of `iso`, `top`, `front`, `right`)
`color`	string		workpiece: material, height, line, deviation; toolpath: feed, force, chip, margin, line, z, limit
`path`	boolean	`false`	
`top`	integer	`15`	profile: lines shown, at most 40
`tol`	number	`0.02`	deviation: the tolerance band, mm
`width`	integer	`960`	
`height`	integer	`640`	
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `image`, `kind`, `of`.

`result_export` - Export the machined part

changes the machine

The workpiece as the machine has cut it, into a file the user can open in other programs (CAD, slicers, viewers) or send on: format stl = a closed binary STL mesh of the material left, in frame program (the program's zero, as CAM draws the part), blank (the blank's front left corner at its lowest point) or machine (machine coordinates); format png = a 16-bit height map seen from above. res: the cells' size, mm (left out: the finest of 0.02 .. 0.5 mm within about a million cells, never finer than the workpiece's own cells). Written to path - a .stl / .png file, or a folder (made if missing) for a name of its own - else into the exports folder (Documents/Twinwright Studio/Exports); an existing file is kept unless overwrite. The same cells as inspect_part and result_compare read. Answers the file's path, size and triangles; after program_run or program_simulate on that machine it is the machined part.

Parameters

Name	Type	Default	Description
`format`	string	`"stl"`	(one of `stl`, `png`)
`frame`	string	`"program"`	(one of `program`, `blank`, `machine`)
`res`	number		cell size, mm (left out: chosen as said)
`path`	string		a .stl / .png file, or a folder; left out: the exports folder
`overwrite`	boolean	`false`	
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `path`, `bytes`, `format`, `frame`, `res_mm`, `triangles`, `removed_mm3`.

`part_save` - Part library: save the workpiece

changes the machine

The machine's workpiece as cut into the user's part library (the same library as the Studio window's File > Part library..., the same record as its "Save to part library..."): a folder in Documents/Twinwright Studio/Parts with workpiece.stl (the machined part, the program's frame), top.png (it from above) and part.json - the program's name, version and SHA-256 (the program itself stays in the programs' folder), the blank, fixture and tool table, the machine, the last run's time, alarms and collisions, and what you add: name (left out: the program's), tags, notes, and the inspection (inspect_part's summary) or comparison (result_compare's) to keep with it. Saved as by you ("made_by": "ai:<your name>"). part_list lists the records.

Parameters

Name	Type	Default	Description
`name`	string		
`tags`	array of string		
`notes`	string		
`inspection`	object		
`comparison`	object		
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `id`, `folder`, `files`, `record`.

`part_export` - Part library: export a record's part

changes the machine

A part library record's machined part (format stl: workpiece.stl, a closed mesh in the program's frame) or its picture from above (png: top.png) copied to path - a .stl / .png file, or a folder (made if missing) - else into the exports folder (Documents/Twinwright Studio/Exports), named after the record; an existing file is kept unless overwrite. For the workpiece on a machine now: result_export.

Parameters

Name	Type	Default	Description
`id` *	string		
`format`	string	`"stl"`	(one of `stl`, `png`)
`path`	string		
`overwrite`	boolean	`false`	

Returns (structured content): `id`, `path`, `bytes`, `format`.

`part_load` - Part library: make another the same

changes the machine

A part library record set up again on a machine as it was cut, as the Studio window's "Make another the same..." does and as an example loads: its blank and fixture, T1 in the spindle, the tool table, homing, the work zero as recorded, and the program - the version it was cut with, its SHA-256 checked - loaded, not started (program_run starts it). The window's machine only with the user's yes there. Refused when that version is not in the programs' folder any more or its text changed; when it would not be as it was cut - another machine, a tool the program calls missing from the recorded table - refused saying what differs, nothing changed: tell the user, and only if they still want it call again with force: true.

Parameters

Name	Type	Default	Description
`id` *	string		
`force`	boolean	`false`	
`machine`	string		which machine. Leave it out for the machine the user sees and means - the one in their Twinwright Studio window ("window"; needs the window open with its AI switch on). "sandbox" (or "sandbox-<name>") is an invisible practice machine of yours, faster than real time (far faster moving, a few times while cutting), shared by the user's AI clients: the user does not see it - use it only to try something out, and say so when you report. No other names. Left out while the window's AI switch is off: "sandbox"

Returns (structured content): `machine`, `notice`, `id`, `name`, `loaded`, `program`, `version`, `zero`, `started`, `differs`.

`log_query` - Audit log

read-only · the same call twice does no more than once

The audit log: who did what, when, to which machine - the user in the window, the AI clients, the machines themselves (alarms, homing, programs started and ended, collisions), the twin host. Newest first; filters: since / until (a time's start, e.g. "2026-09-29T17:5"), category (machine, operator, program, setup, ai, system), action (a prefix: "machine.alarm", "ai.tool"), who (user, ai, system, or a name), machine, text, severity (warn, error); paged. Useful to see what the user did in the window, or what became of a machine.

Parameters

Name	Type	Default	Description
`since`	string		
`until`	string		
`category`	string		(one of `machine`, `operator`, `program`, `setup`, `ai`, `system`)
`action`	string		
`who`	string		
`machine`	string		
`text`	string		
`severity`	string		(one of `warn`, `error`)
`limit`	integer	`30`	
`offset`	integer	`0`	

Returns (structured content): `machine`, `notice`, `records`, `matching`.

The licence

`licence_status` - Licence and edition

read-only · the same call twice does no more than once

This copy's edition and licence: the edition in force (in_force: learning, maker, pro, shop; classroom, trial), the licence files in the computer's licence folder - each one's kind, holder, seats, dates (valid_until, renew_after, updates_until), days left (a subscription past its period: its days of grace) and whether it holds, and why not - the local trials used, whether a bought licence's updates cover this release, every paid feature with the edition it needs, why it is closed and whether this release has it at all (shipped false: not built yet - never offer it), and the site's pages for the next step (pricing: buy or upgrade; licence: a year more of updates, managing a subscription, offline activation). Changes nothing; gives no activation code and no computer fingerprint. Use it to explain a refusal ("edition: ... comes with the pro edition") and what the user can do.

No parameters.

Returns (structured content): `edition`, `in_force`, `files`, `trials_used`, `features`, `links`.

The part library

`part_list` - Part library: the records

read-only · the same call twice does no more than once

The user's part library: machined parts kept with what made them (Documents/Twinwright Studio/Parts/<id>/), the newest first - each record's id, name, created, made_by (user or ai:<client>), the machine, the program (name, version, SHA-256), the setup, material, tools, results (removed volume, machine time, chatter, collisions, alarms), the inspection and the comparison with the design, files (the machined workpiece as an STL in the program's coordinates, a picture), tags and notes. Filters: query (in the name, notes or program), material, program, tag, since (an ISO time's start, e.g. "2026-10-12"); limit (default 50). deleted: the trash instead (kept 30 days). Changes nothing.

Parameters

Name	Type	Default	Description
`query`	string		
`material`	string		
`program`	string		
`tag`	string		
`since`	string		
`limit`	integer	`50`	
`deleted`	boolean	`false`	

Returns (structured content): `parts`, `matched`, `total`, `folder`.

`part_get` - Part library: one record

read-only · the same call twice does no more than once

One record of the part library (part_list gives the ids): all it keeps, and the paths of its files on this computer (the workpiece's STL, its picture, the program as it was cut). Changes nothing.

Parameters

Name	Type	Default	Description
`id` *	string		

Returns (structured content): `id`, `record`, `paths`.

`part_update` - Part library: name, tags, notes

changes the machine · the same call twice does no more than once

A record's name, tags (a list of words: replaces them) or notes ("" clears them). Nothing else of a record changes: it says what was made, and how.

Parameters

Name	Type	Default	Description
`id` *	string		
`name`	string		
`tags`	array of string		
`notes`	string		

Returns (structured content): `id`.

`part_delete` - Part library: delete or restore

changes the machine (destructive)

A record into the part library's trash - only with the user's yes in the Studio window, asked whatever the settings say (the window not open: refused - the user deletes it there, File > Part library...). It stays in the trash 30 days; restore: true brings it back (no question). part_list {deleted: true} lists the trash.

Parameters

Name	Type	Default	Description
`id` *	string		
`restore`	boolean	`false`	

Returns (structured content): `id`, `deleted`, `restored`.

Reports to support

`report_draft` - Report a problem: draft it for the user

changes the machine

Drafts a report to Twinwright Studio's support (a problem, how to use something, an idea) and opens it in the user's Studio window (Help > Report a problem..., marked as drafted by you). Only the user sends it - you cannot - and only the user attaches the diagnostics package or a picture of the window. Write what happened as the user would: what was done, expected, seen (the version, the system and the audit log's last records go with it anyway). No Studio window open: nothing is shown - tell the user to report it there.

Parameters

Name	Type	Default	Description
`kind`	string	`"problem"`	(one of `problem`, `howto`, `idea`)
`text` *	string		what happened (at most 4000 bytes)

Returns (structured content): `shown`.

`report_list` - My reports: their state and replies

read-only · the same call twice does no more than once

The reports sent from Studio on this computer (Help > My reports) as the window last saw them: each ticket's id ("TW-1042"), title, kind, status (new, in_progress, answered, solved), when it last changed, the replies not read yet, and its conversation as the window last fetched it (messages: from you or support, at, text, attachments). id: that one only. Read-only: nothing is asked of the support service.

Parameters

Name	Type	Default	Description
`id`	string		

Returns (structured content): `reports`, `count`.

The user's window

`window_show` - Show the user something in the Studio window

changes the machine · the same call twice does no more than once

As you guide the user, show them in their own Studio window what you mean: a workspace (setup, cam, program, run, analysis, diagnostics), the 3D view's side (iso, front, right, top), the tool tip's trail, the program's path and the tool's close-up on or off, the trail's colouring (plain, feed, force, chip, margin, line) and the workpiece's (material, deviation, line), the dock's page (program, curves, console, messages, tools, ai; in the CAM workspace checks, versions, preview; in Setup inspect), a step of the Setup panel (power, homing, workpiece, tools, zero, program, cut). The window does it as the user's own clicks would: only what it shows changes - nothing on the machine, no yes asked. Give one or more; they are done in that order (the workspace first).

Parameters

Name	Type	Default	Description
`workspace`	string		(one of `setup`, `cam`, `program`, `run`, `analysis`, `diagnostics`)
`view`	string		(one of `iso`, `front`, `right`, `top`)
`trail`	boolean		
`path`	boolean		
`close_up`	boolean		
`trail_colour`	string		(one of `plain`, `feed`, `force`, `chip`, `margin`, `line`)
`stock_colour`	string		(one of `material`, `deviation`, `line`)
`dock`	string		(one of `program`, `curves`, `console`, `messages`, `tools`, `ai`, `checks`, `versions`, `preview`, `inspect`)
`setup_step`	string		(one of `power`, `homing`, `workpiece`, `tools`, `zero`, `program`, `cut`)

Returns (structured content): `shown`, `what`.

`settings` - The window's settings: read, or change with the user's yes

changes the machine · the same call twice does no more than once

The settings of the user's Studio an AI may read and change: language ("auto" or a language pack's tag such as "zh-CN"), region (how numbers, money and dates look: "auto", "lang" or such as "de-DE"), units ("auto", "metric", "imperial" ...), currency (ISO 4217, "": the region's), currency_symbol and machine_rate (the machine-hour rate for cost estimates, in that currency; 0: none). Without arguments: their values and the choices. With some: the user says yes in the window, then it applies them as its own Settings' Apply. The others (your own access, privacy, the log, the look) only the user, in Tools > Settings.

Parameters

Name	Type	Default	Description
`language`	string		
`region`	string		
`units`	string		
`currency`	string		
`currency_symbol`	string		
`machine_rate`	number		(at least 0)

Returns (structured content): `settings`, `choices`, `set`, `what`.

The manual

`manual_search` - Search Studio's manual

read-only · the same call twice does no more than once

The manual Studio has on this computer (the user manual, the operations manual, the technical manual, the interface reference - what Help > Documentation shows), searched as the window searches it: every section with all of q's words (any case). Each hit: its chapter, where it is (the breadcrumb), its page (what Help > Documentation opens) and a snippet with the word found between **. lang: the manual's language (left out: the interface's; one it has not: English). Read-only; answer the user from it rather than from memory.

Parameters

Name	Type	Default	Description
`q` *	string		the words to find
`lang`	string		such as zh-CN, de, ja
`max`	integer	`8`	(at least 1; at most 30)

Returns (structured content): `lang`, `places`, `pages`, `hits`.

Einstellungsreferenz

Alle Einstellungen der Steuerung. `$<n>` zeigt eine an, `$<n>=<value>` ändert sie, und `$$` listet alle auf. Die angegebenen Werte sind die der MC-1. Eine mit *Neustart* markierte Einstellung wird erst nach einem Neustart der Steuerung wirksam.

Inhalt

- Allgemein
- Steuersignale
- Endschalter
- Kühlung
- Spindel
- Werkzeugwechsel
- Referenzfahrt
- Antasten
- Parken/Schutztür
- Tippbetrieb
- Modbus
- Schrittmotor
- Frequenzumrichter
- X-Achse
- Y-Achse
- Z-Achse

Allgemein

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
`$10`	Statusbericht-Optionen		Bitfeld		15871	Was der Echtzeit-Statusbericht enthält. Der Parserzustand wird als eigene Zeile gesendet, und zwar nur, wenn er sich geändert hat. Bits: 0 Position in Maschinenkoordinaten, 1 Pufferzustand, 2 Satznummern, 3 Vorschub und Drehzahl, 4 Pin-Zustand, 5 Nullpunktverschiebung (WCO), 6 Overrides, 7 Antastkoordinaten, 8 Puffer synchronisieren bei WCO-Änderung, 9 Parserzustand, 10 Alarm-Unterstatus, 11 Lauf-Unterstatus, 12 Bei Referenzfahrt aktivieren, 13 Restweg
`$11`	Eckenabweichung	mm	Dezimalzahl		0.002	Wie weit die Bahn an einer Ecke zwischen zwei Bewegungen abweichen darf; das bestimmt die Geschwindigkeit in der Ecke. Kleiner bedeutet langsamer und genauer.
`$12`	Kreisbogentoleranz	mm	Dezimalzahl		0.002	Wie genau Kreisbögen abgefahren werden.
`$13`	Berichte in Zoll		boolesch		0	Positionen und Geschwindigkeiten in Berichten werden in Zoll angegeben.
`$28`	G73-Rückzugsweg	mm	Dezimalzahl		0.100	Der kurze Rückzug zwischen den Bohrschritten von G73.
`$32`	Betriebsart		Auswahl		0	Fräsmaschine, Laser oder Drehmaschine. Auswahl: 0 Normal, 1 Lasermodus, 2 Drehmaschinenmodus
`$39`	Herkömmliche Echtzeitbefehle aktivieren		boolesch		1	„?“, „!“ und „~“ gelten überall als Echtzeitbefehle, auch innerhalb von $-Befehlen und in Kommentaren.
`$60`	Overrides zurücksetzen		boolesch		0	Die Overrides gehen am Programmende auf 100 % zurück.
`$62`	Ruhezustand aktivieren		boolesch		0	Nach langer Leerlaufzeit in den Ruhezustand wechseln.
`$63`	Aktionen bei Vorschub halt		Bitfeld		3	Was bei Vorschub halt außer dem Anhalten der Bewegung noch geschieht. Bits: 0 Laser bei Halt ausschalten, 1 Spindel- und Kühlungszustand beim Fortsetzen wiederherstellen, 2 Drehzahl auf Minimum setzen (außer bei Laser-M4)
`$64`	Alarm beim Start erzwingen		boolesch		0	Nach dem Einschalten im Alarmzustand starten.
`$384`	G92-Speicherung deaktivieren		boolesch		0	G92-Verschiebungen bleiben über ein Aus- und Wiedereinschalten nicht erhalten.
`$398`	Sätze im Planerpuffer *(Neustart)*		Ganzzahl	30–1000	100	Bewegungen, über die der Planer vorausschaut.
`$481`	Intervall für automatische Berichte *(Neustart)*	ms	Ganzzahl	100–1000	0	Statusberichte in diesem Intervall ungefragt senden (0: aus).
`$484`	Entsperren nach Not-Halt erforderlich		boolesch		1	Nach einem Not-Halt bleibt die Maschine gesperrt, bis sie entsperrt ($X) oder referenziert wird.
`$486`	Koordinatensysteme sperren		Bitfeld		0	Koordinatensysteme, die G10 nicht ändern darf. Bits: 0 G59.1, 1 G59.2, 2 G59.3
`$676`	Aktionen bei Reset		Bitfeld		15	Was ein Reset löscht. Bits: 0 Referenzstatus löschen bei Positionsverlust, 1 Verschiebungen löschen (außer G92), 2 Eilgang-Override zurücksetzen, 3 Vorschub-Override zurücksetzen

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$1030	Auftrag stoppen bei Verlust des Hosts		boolesch		1	Wenn die Verbindung des Sender-Programms, das einen Auftrag ausführt, 0.1 s lang unterbrochen ist (USB abgezogen, Port geschlossen), stoppt der Auftrag wie bei einem Stopp (0x19): Die Bewegung wird bis zum Stillstand abgebremst, Spindel und Kühlung werden ausgeschaltet, die vom Sender-Programm gesendeten Sätze, die noch nicht ausgeführt wurden, werden verworfen, die Position bleibt erhalten. Das Sender-Programm wird benachrichtigt, sobald es wieder verbunden ist. Ein Programm aus dem Dateisystem läuft auch ohne sein Sender-Programm weiter.

Steuersignale

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$14	Steuereingänge invertieren		Bitfeld		0	Steuereingänge, die invertiert sind. Bits: 1 Vorschub halt, 2 Zyklusstart, 6 Not-Halt, 8 Motorstörung
$17	Pull-up der Steuereingänge deaktivieren		Bitfeld		0	Steuereingänge ohne internen Pull-up-Widerstand. Bits: 1 Vorschub halt, 2 Zyklusstart, 6 Not-Halt, 8 Motorstörung

Endschalter

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$5	Endschaltereingänge invertieren		Achsmaske		7	Endschaltereingänge, die invertiert sind, z. B. für Öffner (NC).
$18	Pull-up der Endschaltereingänge deaktivieren		Achsmaske		0	Endschaltereingänge ohne internen Pull-up-Widerstand.
$20	Software-Endschalter aktivieren		boolesch		1	Bewegungen über den Verfahrweg ($130 ...) hinaus ablehnen. Erfordert eine Referenzfahrt, damit die Position bekannt ist.
$21	Hardware-Endschalter aktivieren		Bitfeld		1	Sofort anhalten, wenn ein Endschalter auslöst. Strenger Modus: keinerlei Bewegung, solange ein Schalter aktiv ist. Bits: 0 Aktivieren, 1 Strenger Modus (Bit 0 aktiviert die übrigen)

Kühlung

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$15	Kühlungsausgänge invertieren		Bitfeld		0	Kühlungsausgänge, die Low-aktiv sind. Bits: 0 Flutkühlung, 1 Sprühkühlung
$673	Einschaltverzögerung der Kühlung	s	Dezimalzahl	0.5–20	0.0	Zeit, die die Kühlung nach M7/M8 zum Anlaufen bekommt.

Spindel

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$30	Maximale Spindeldrehzahl	U/min	Dezimalzahl		24000.000	Die höchste Spindeldrehzahl.
$31	Minimale Spindeldrehzahl	U/min	Dezimalzahl		6000.000	Die niedrigste Spindeldrehzahl.
$340	Toleranz für „Spindeldrehzahl erreicht“	%	Dezimalzahl		5.0	Bewegungen nach M3/M4 warten, bis die Spindeldrehzahl höchstens um so viele Prozent von der programmierten abweicht (0: nicht warten).
$394	Einschaltverzögerung der Spindel	s	Dezimalzahl	0.5–60	15.0	Zeit, die die Spindel nach M3/M4 zum Hochlaufen auf die Drehzahl bekommt (mit $340: die längste Wartezeit vor einem Alarm).
$539	Ausschaltverzögerung der Spindel	s	Dezimalzahl	0.5–60	0.0	Zeit, die die Spindel nach M5 zum Stillstand bekommt.

Werkzeugwechsel

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$341	Werkzeugwechselmodus		Auswahl		3	Was M6 bewirkt. Auswahl: 0 Normal, 1 Manuelle Werkzeugvermessung, 2 Manuelle Werkzeugvermessung bei G59.3, 3 Automatische Werkzeugvermessung bei G59.3, 4 M6 ignorieren
$342	Antastweg beim Werkzeugwechsel	mm	Dezimalzahl		70.0	Wie weit nach dem Werkzeuglängensensor gesucht wird.
$343	Feinvorschub beim Werkzeugwechsel	mm/min	Dezimalzahl		25.0	Die langsame Geschwindigkeit, mit der die Position des Werkzeuglängensensors genau bestimmt wird.
$344	Suchgeschwindigkeit beim Werkzeugwechsel	mm/min	Dezimalzahl		200.0	Die schnelle Geschwindigkeit, mit der nach dem Werkzeuglängensensor gesucht wird.

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$345	Freifahrgeschwindigkeit vom Messtaster beim Werkzeugwechsel	mm/min	Dezimalzahl		200.0	Die Geschwindigkeit, mit der vom Werkzeuglängensensor freigefahren wird.
$346	Werkzeugwechsel-Optionen		Bitfeld		0	Optionen für den Werkzeugwechsel. Bits: 0 Position nach M6 wiederherstellen, 1 Werkzeug bei G30 wechseln, 2 Schnelles Freifahren vom Messtaster
$485	Werkzeugnummer nach Neustart beibehalten		boolesch		0	Die Nummer des Werkzeugs in der Spindel bleibt über ein Aus- und Wiedereinschalten erhalten.

Referenzfahrt

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$22	Referenzfahrtzyklus		Bitfeld		541	Der Referenzfahrtzyklus und seine Optionen. Bits: 0 Aktivieren, 1 Einzelachsbefehle aktivieren, 2 Referenzfahrt beim Start erforderlich, 3 Maschinennullpunkt auf 0 setzen, 4 Zwei Schalter teilen sich einen Eingang, 5 Manuell erlauben, 6 Override-Sperren, 9 Vorschübe je Achse, 10 Startsätze nur nach abgeschlossener Referenzfahrt ausführen (Bit 0 aktiviert die übrigen)
$23	Referenzfahrtrichtung invertieren		Achsmaske		0	Achsen, die in Richtung ihres negativen Endes referenzieren.
$26	Entprellzeit der Referenzschalter	Millisekunden	Ganzzahl		250	Wartezeit nach dem Auslösen eines Schalters, bevor die nächste Bewegung der Referenzfahrt beginnt.
$27	Freifahrweg der Referenzschalter	mm	Dezimalzahl		2.000	Wie weit die Referenzfahrt vom Schalter zurückfährt.
$43	Durchgänge der Referenzfahrt		Ganzzahl	1–128	1	Wie oft der Schaltpunkt bestimmt wird.
$44	Achsen der Referenzfahrt, erste Phase		Achsmaske		4	Achsen, die zuerst referenziert werden.
$45	Achsen der Referenzfahrt, zweite Phase		Achsmaske		3	Achsen, die an zweiter Stelle referenziert werden.
$46	Achsen der Referenzfahrt, dritte Phase		Achsmaske		0	Achsen, die an dritter Stelle referenziert werden.

Antasten

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$6	Messtastereingang invertieren		Bitfeld		1	Invertiert den Messtastereingang, z. B. für einen Werkzeuglängensensor als Öffner (NC). Bits: 0 Messtaster
$19	Pull-up des Messtastereingangs deaktivieren		Bitfeld		0	Der Messtastereingang ohne internen Pull-up-Widerstand. Bits: 0 Messtaster
$65	Antast-Optionen		Bitfeld		0	Wie Antastbewegungen behandelt werden. Bits: 0 Vorschub-Override zulassen, 1 Software-Endschalter anwenden

Parken/Schutztür

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$41	Parkzyklus		Bitfeld		0	Die Spindel zurückziehen, wenn die Schutztür geöffnet wird. Bits: 0 Aktivieren, 1 Beim Start deaktivieren, 2 Park-Override-Steuerung aktivieren (Bit 0 aktiviert die übrigen)
$42	Parkachse		Auswahl		2	Die Achse, die parkt. Auswahl: 0 X, 1 Y, 2 Z
$56	Langsamer Rückzugsweg beim Parken	mm	Dezimalzahl		5.0	Wie weit das Werkzeug zuerst langsam herausgezogen wird.
$57	Langsame Rückzugsgeschwindigkeit beim Parken	mm/min	Dezimalzahl		100.0	Die Geschwindigkeit dieses Herausziehens.
$58	Parkziel	mm	Dezimalzahl	≥ -100000	-5.0	Wohin die Parkachse fährt, in Maschinenkoordinaten.

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$59	Schnelle Parkgeschwindigkeit	mm/min	Dezimalzahl		500.0	Die Geschwindigkeit des restlichen Teils der Parkbewegung.
$61	Schutztür-Optionen		Bitfeld		0	Was eingeschaltet bleibt, solange die Schutztür offen ist. Bits: 1 Kühlungszustand bei geöffneter Tür beibehalten
$392	Einschaltverzögerung der Spindel	s	Dezimalzahl	0.5–60	4.0	Zeit, die die Spindel zum Hochlaufen auf die Drehzahl bekommt, wenn die Tür wieder geschlossen wird.
$393	Einschaltverzögerung der Kühlung	s	Dezimalzahl	0.5–20	1.0	Zeit, die die Kühlung zum Anlaufen bekommt, wenn die Tür wieder geschlossen wird.

Tippbetrieb

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$40	Tippbefehle begrenzen		boolesch		0	Tippbewegungen halten an den Software-Endschaltern an, statt abgelehnt zu werden.

Modbus

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$374	Modbus-Baudrate		Auswahl		3	Baudrate der Modbus-Leitung (die des FU muss übereinstimmen). Auswahl: 0 2400, 1 4800, 2 9600, 3 19200, 4 38400, 5 115200
$375	Modbus-Empfangs-Timeout	Millisekunden	Ganzzahl	50–250	50	Wie lange auf eine Modbus-Antwort gewartet wird.
$681	Serielles Modbus-Format		Auswahl		0	Parität der Modbus-Leitung (die des FU muss übereinstimmen). Auswahl: 0 8 Bit, keine Parität, 1 8 Bit, gerade Parität, 2 8 Bit, ungerade Parität

Schrittmotor

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$0	Schrittimpulsdauer	Mikrosekunden	Dezimalzahl	≥ 2.0	5.0	Länge der Schrittimpulse in Mikrosekunden. Die Antriebe brauchen ein Minimum (siehe deren Handbuch; die Closed-Loop-Antriebe der MC-1 brauchen 2.5).
$1	Leerlaufverzögerung der Schrittmotoren	Millisekunden	Ganzzahl	≤ 65535	255	Wie lange die Antriebe nach dem Ende der Bewegung freigegeben bleiben, damit die Achsen zur Ruhe kommen, bevor der Strom abgeschaltet wird. 255 hält sie dauerhaft freigegeben.
$2	Schrittimpuls invertieren		Achsmaske		0	Schrittausgänge, die Low-aktiv sind.
$3	Schrittrichtung invertieren		Achsmaske		5	Achsen, deren Richtungsausgang invertiert ist (sonst läuft die Achse in die falsche Richtung).
$4	Freigabeausgänge der Schrittmotoren invertieren		Achsmaske		7	Freigabeausgänge der Antriebe, die Low-aktiv sind (die meisten Antriebe). Bei nur einem gemeinsamen Freigabesignal zählt nur X.
$29	Schrittimpuls-Verzögerung	Mikrosekunden	Dezimalzahl	≤ 20	5.0	Zeit zwischen dem Setzen der Richtungsausgänge und dem Schrittimpuls.
$37	Dauerhaft freigegebene Schrittmotoren		Achsmaske		7	Antriebe, die freigegeben bleiben, wenn die Maschine bereit ist oder einen Alarm hat.
$680	Freigabeverzögerung der Schrittmotoren	ms	Ganzzahl	≤ 500	0	Zeit zwischen dem Freigeben der Antriebe und dem ersten Schritt.

Frequenzumrichter

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$460	Modbus-Adresse des Spindel-Frequenzumrichters		Ganzzahl	≤ 255	1	Die Modbus-Adresse des FU.

X-Achse

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
$100	Wegauflösung X-Achse	Schritte/mm	Dezimalzahl		800.00000	Schritte pro Millimeter der Achse: Schritte pro Motorumdrehung mal Mikroschritte, geteilt durch die Steigung der Gewindespindel.
$110	Höchstgeschwindigkeit X-Achse	mm/min	Dezimalzahl		3000.000	Die höchste Geschwindigkeit der Achse, auch für Eilgänge.
$120	Beschleunigung X-Achse	mm/s²	Dezimalzahl		300.000	Die Beschleunigung der Achse.

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
`$130`	Maximaler Verfahrweg X-Achse	mm	Dezimalzahl		202.000	Der Verfahrweg der Achse ab ihrem Referenzpunkt, für die Software-Endschalter.
`$180`	Referenzfahrt-Feinvorschub X-Achse	mm/min	Dezimalzahl		50.0	Die langsame Geschwindigkeit, mit der der Schaltpunkt der Achse genau bestimmt wird.
`$190`	Referenzfahrt-Suchgeschwindigkeit X-Achse	mm/min	Dezimalzahl		1200.0	Die schnelle Geschwindigkeit, mit der nach dem Schalter der Achse gesucht wird.
`$1000`	Input-Shaper X-Achse		Auswahl		0	Formt die Bewegung der Achse so, dass sie keine Resonanz der Maschine bei der unten eingestellten Frequenz anregt: ZV ist bei dieser Frequenz exakt, die anderen unterdrücken ein breiteres Band, aber weniger exakt. Die Bewegung kommt bis zu 1.5 Perioden später zur Ruhe. Auswahl: 0 Ohne, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI
`$1010`	Input-Shaper-Frequenz X-Achse	Hz	Dezimalzahl	15–500	100.0	Die Resonanz, die der Input-Shaper der Achse unterdrückt.
`$1020`	Input-Shaper-Dämpfungsgrad X-Achse		Dezimalzahl	≤ 0.5	0.100	Der Dämpfungsgrad dieser Resonanz.

Y-Achse

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
`$101`	Wegauflösung Y-Achse	Schritte/mm	Dezimalzahl		800.00000	Schritte pro Millimeter der Achse: Schritte pro Motorumdrehung mal Mikroschritte, geteilt durch die Steigung der Gewindespindel.
`$111`	Höchstgeschwindigkeit Y-Achse	mm/min	Dezimalzahl		3000.000	Die höchste Geschwindigkeit der Achse, auch für Eilgänge.
`$121`	Beschleunigung Y-Achse	mm/s²	Dezimalzahl		300.000	Die Beschleunigung der Achse.
`$131`	Maximaler Verfahrweg Y-Achse	mm	Dezimalzahl		152.000	Der Verfahrweg der Achse ab ihrem Referenzpunkt, für die Software-Endschalter.
`$181`	Referenzfahrt-Feinvorschub Y-Achse	mm/min	Dezimalzahl		50.0	Die langsame Geschwindigkeit, mit der der Schaltpunkt der Achse genau bestimmt wird.
`$191`	Referenzfahrt-Suchgeschwindigkeit Y-Achse	mm/min	Dezimalzahl		1200.0	Die schnelle Geschwindigkeit, mit der nach dem Schalter der Achse gesucht wird.
`$1001`	Input-Shaper Y-Achse		Auswahl		0	Formt die Bewegung der Achse so, dass sie keine Resonanz der Maschine bei der unten eingestellten Frequenz anregt: ZV ist bei dieser Frequenz exakt, die anderen unterdrücken ein breiteres Band, aber weniger exakt. Die Bewegung kommt bis zu 1.5 Perioden später zur Ruhe. Auswahl: 0 Ohne, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI
`$1011`	Input-Shaper-Frequenz Y-Achse	Hz	Dezimalzahl	15–500	100.0	Die Resonanz, die der Input-Shaper der Achse unterdrückt.
`$1021`	Input-Shaper-Dämpfungsgrad Y-Achse		Dezimalzahl	≤ 0.5	0.100	Der Dämpfungsgrad dieser Resonanz.

Z-Achse

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
`$102`	Wegauflösung Z-Achse	Schritte/mm	Dezimalzahl		800.00000	Schritte pro Millimeter der Achse: Schritte pro Motorumdrehung mal Mikroschritte, geteilt durch die Steigung der Gewindespindel.
`$112`	Höchstgeschwindigkeit Z-Achse	mm/min	Dezimalzahl		1500.000	Die höchste Geschwindigkeit der Achse, auch für Eilgänge.
`$122`	Beschleunigung Z-Achse	mm/s²	Dezimalzahl		200.000	Die Beschleunigung der Achse.
`$132`	Maximaler Verfahrweg Z-Achse	mm	Dezimalzahl		82.000	Der Verfahrweg der Achse ab ihrem Referenzpunkt, für die Software-Endschalter.
`$182`	Referenzfahrt-Feinvorschub Z-Achse	mm/min	Dezimalzahl		50.0	Die langsame Geschwindigkeit, mit der der Schaltpunkt der Achse genau bestimmt wird.
`$192`	Referenzfahrt-Suchgeschwindigkeit Z-Achse	mm/min	Dezimalzahl		600.0	Die schnelle Geschwindigkeit, mit der nach dem Schalter der Achse gesucht wird.
`$1002`	Input-Shaper Z-Achse		Auswahl		0	Formt die Bewegung der Achse so, dass sie keine Resonanz der Maschine bei der unten eingestellten Frequenz anregt: ZV ist bei dieser Frequenz exakt, die anderen unterdrücken ein breiteres Band, aber weniger exakt. Die Bewegung kommt bis zu 1.5 Perioden später zur Ruhe. Auswahl: 0 Ohne, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI

Einstellung	Name	Einheit	Typ	Bereich	MC-1	Beschreibung
`$1012`	Input-Shaper-Frequenz Z-Achse	Hz	Dezimalzahl	15–500	100.0	Die Resonanz, die der Input-Shaper der Achse unterdrückt.
`$1022`	Input-Shaper-Dämpfungsgrad Z-Achse		Dezimalzahl	≤ 0.5	0.100	Der Dämpfungsgrad dieser Resonanz.

Alarmcodes

Die Alarme der Steuerung. Ein Alarm hält die Maschine an und sperrt sie, bis Sie ihn aufheben: `$X` oder Referenzfahrt, wie es der Alarm angibt. `$EA` listet sie auf der Steuerung selbst auf.

Code	Bedeutung
`ALARM:1`	Hardware-Endschalter: Während der Bewegung wurde ein Endschalter ausgelöst. Die Position ist wahrscheinlich verloren; referenzieren Sie erneut
`ALARM:2`	Software-Endschalter: Das Bewegungsziel liegt außerhalb des Verfahrwegs der Maschine. Die Position bleibt erhalten; Sie können entsperren
`ALARM:3`	Reset oder Not-Halt während der Bewegung. Die Position ist wahrscheinlich verloren; referenzieren Sie erneut
`ALARM:4`	Messtaster: Der Messtaster hatte beim Start des Antastzyklus nicht den erwarteten Zustand
`ALARM:5`	Messtaster: kein Kontakt innerhalb des programmierten Verfahrwegs
`ALARM:6`	Referenzfahrt: Die Referenzfahrt wurde durch einen Reset abgebrochen
`ALARM:7`	Referenzfahrt: Die Schutztür wurde während der Referenzfahrt geöffnet
`ALARM:8`	Referenzfahrt: Der Endschalter hat nach dem Freifahren nicht freigegeben. Prüfen Sie den Freifahrweg ($27) und die Verdrahtung
`ALARM:9`	Referenzfahrt: Innerhalb des Suchwegs wurde kein Endschalter gefunden. Prüfen Sie die Einstellungen zum Verfahrweg und die Verdrahtung
`ALARM:10`	Not-Halt. Entriegeln Sie ihn und führen Sie dann einen Reset aus
`ALARM:11`	Referenzfahrt erforderlich: Führen Sie die Referenzfahrt aus ($H)
`ALARM:12`	Ein Endschalter ist betätigt. Fahren Sie vom Endschalter frei, bevor Sie fortfahren
`ALARM:13`	Messtasterschutz: Der Messtaster wurde außerhalb eines Antastzyklus ausgelöst
`ALARM:14`	Die Spindel hat ihre Drehzahl nicht rechtzeitig erreicht
`ALARM:15`	Referenzfahrt: Der zweite Schalter einer Achse mit automatischem Rechtwinkelabgleich wurde nicht gefunden
`ALARM:16`	Der Einschalt-Selbsttest ist fehlgeschlagen
`ALARM:17`	Ein Motorantrieb meldet eine Störung
`ALARM:18`	Referenzfahrt: Die Einstellungen der Referenzfahrt sind ungültig
`ALARM:19`	Modbus: keine oder eine fehlerhafte Antwort eines Geräts (z. B. des FU)
`ALARM:20`	Die E/A-Erweiterung antwortet nicht
`ALARM:21`	Der nichtflüchtige Speicher (Einstellungsspeicher) ist ausgefallen
`ALARM:22`	Ein Puffer ist übergelaufen

Fehlercodes

Die Antwort `error:N` auf einen Satz, den die Steuerung abgelehnt hat. Der Satz hat nichts bewirkt. `$EE` listet sie auf der Steuerung selbst auf.

Code	Bedeutung
`error:1`	Einem G-Code-Wort fehlt der Buchstabe: Jedes Wort besteht aus einem Buchstaben, gefolgt von einer Zahl
`error:2`	Eine Zahl fehlt oder ist keine gültige Zahl
`error:3`	Unbekannter oder nicht unterstützter „$"-Befehl
`error:4`	Ein negativer Wert wurde angegeben, wo nur positive Werte erlaubt sind
`error:5`	Die Referenzfahrt ist in den Einstellungen nicht aktiviert ($22)
`error:6`	Der Schrittimpuls muss mindestens 2 Mikrosekunden lang sein
`error:7`	Gespeicherte Einstellungen konnten nicht gelesen werden; die betroffenen Einstellungen wurden auf ihre Standardwerte gesetzt
`error:8`	Dieser „$"-Befehl wird nur angenommen, solange die Maschine bereit ist
`error:9`	G-Code wird während eines Alarms oder im Tippbetrieb nicht angenommen
`error:10`	Software-Endschalter erfordern eine Referenzfahrt: Aktivieren Sie zuerst die Referenzfahrt ($22)
`error:11`	Der Satz war länger, als die Steuerung annimmt, und wurde nicht ausgeführt
`error:12`	Der Wert würde zu einer höheren Schrittrate führen, als die Steuerung erzeugen kann
`error:13`	Die Schutztür ist offen
`error:14`	Der Text ist zu lang, um gespeichert zu werden (Startsatz oder Build-Info)
`error:15`	Das Tippziel liegt außerhalb des Verfahrwegs der Maschine; die Tippbewegung wurde nicht ausgeführt
`error:16`	Ein Tippbefehl braucht „=" und darf nur Worte enthalten, die zum Tippen gehören
`error:17`	Der Lasermodus erfordert einen PWM-Spindelausgang
`error:18`	Die Steuerung wurde zurückgesetzt
`error:19`	Der Wert muss größer als null sein
`error:20`	Der Satz enthält einen nicht unterstützten oder ungültigen G-Code-Befehl
`error:21`	Zwei Befehle derselben modalen Gruppe in einem Satz
`error:22`	Es wurde noch kein Vorschub gesetzt
`error:23`	Dieser Befehl erfordert eine ganze Zahl
`error:24`	Zwei Befehle im Satz beanspruchen beide die Achsworte
`error:25`	Ein Wort kommt im Satz zweimal vor
`error:26`	Der Befehl braucht Achsworte, und der Satz hat keine
`error:27`	Ungültige Satznummer
`error:28`	Ein vom Befehl benötigtes Wertwort fehlt
`error:29`	G59.1, G59.2 und G59.3 werden nicht unterstützt
`error:30`	G53 funktioniert nur mit G0 und G1
`error:31`	Der Satz enthält Achsworte, die kein Befehl verwendet
`error:32`	Ein Kreisbogen (G2/G3) braucht mindestens ein Achswort in seiner Ebene
`error:33`	Das Bewegungsziel ist ungültig
`error:34`	Der Kreisbogenradius ist ungültig
`error:35`	Ein Kreisbogen (G2/G3) braucht in seiner Ebene mindestens einen Mittelpunktsabstand (I, J oder K)
`error:36`	Der Satz enthält Wertworte, die kein Befehl verwendet
`error:37`	G43.1 benötigt die konfigurierte Werkzeuglängenachse
`error:38`	Die Werkzeugnummer ist höher als unterstützt, oder dieses Werkzeug gibt es nicht
`error:39`	Ein Wert liegt außerhalb des Bereichs
`error:40`	Nicht erlaubt, solange ein Werkzeugwechsel aussteht
`error:41`	Für diese Bewegung muss die Spindel laufen (konstante Schnittgeschwindigkeit oder Spindelsynchronisation)
`error:42`	Gewindeschneiden erfordert die ZX-Ebene (G18)
`error:43`	Der Vorschub ist höher als das Maximum
`error:44`	Die Spindeldrehzahl liegt außerhalb des Bereichs
`error:45`	Ein Endschalter ist betätigt: Es ist nur eine Referenzfahrt erlaubt

Code	Bedeutung
`error:46`	Referenzieren Sie die Maschine zuerst ($H)
`error:47`	Das aktuelle Werkzeug ist nicht bekannt: Setzen Sie es mit M61
`error:48`	Widersprüchliche Wertworte
`error:49`	Der Einschalt-Selbsttest ist fehlgeschlagen: Schalten Sie die Steuerung aus und wieder ein
`error:50`	Der Not-Halt ist aktiv
`error:51`	Ein Motorantrieb meldet eine Störung
`error:52`	Der Einstellwert liegt außerhalb des Bereichs
`error:53`	Diese Einstellung gibt es bei dieser Steuerung nicht
`error:54`	Die Rückzugsposition liegt unter der Bohrtiefe
`error:55`	Zwei Achsen mit automatischem Rechtwinkelabgleich können nicht gemeinsam referenziert werden
`error:56`	Dieses Koordinatensystem ist gesperrt ($486)
`error:57`	Unerwartete Marke für Programmanfang/-ende (%)
`error:58`	Der Port ist nicht verfügbar
`error:60`	Die SD-Karte konnte nicht eingebunden werden
`error:62`	Das Verzeichnis konnte nicht aufgelistet werden
`error:63`	Verzeichnis nicht vorhanden
`error:64`	Die SD-Karte ist nicht eingebunden
`error:65`	Kein Dateisystem eingebunden (keine Karte?)
`error:66`	Das Dateisystem ist schreibgeschützt
`error:71`	Ausdruck: unbekannter Operator oder unbekannte Funktion
`error:72`	Ausdruck: Division durch null
`error:73`	Ausdruck: Argument außerhalb des Bereichs
`error:74`	Ausdruck: ungültiges Argument
`error:75`	Ausdruck: Syntaxfehler
`error:76`	Ausdruck: Das Ergebnis ist keine Zahl
`error:77`	Authentifizierung erforderlich
`error:78`	Zugriff verweigert
`error:79`	Nicht erlaubt, solange ein kritisches Ereignis aktiv ist
`error:80`	Diese O-Wort-Anweisung braucht ein Programm, das die Steuerung erneut lesen kann (aus ihrem Dateisystem)
`error:81`	Unbekannte oder falsch platzierte O-Wort-Anweisung
`error:82`	Zu viele gleichzeitig offene O-Wort-Blöcke oder -Aufrufe
`error:83`	Kein Platz mehr für die O-Wort-Anweisung
`error:84`	Die Datei konnte nicht geöffnet werden
`error:85`	Das Dateisystem konnte nicht formatiert werden
`error:86`	Der Port kann nicht verwendet werden
`error:87`	In der Spindel ist ein Werkzeug
`error:88`	In der Spindel ist kein Werkzeug
`error:89`	Die Datei konnte nicht gelöscht werden
`error:253`	Benutzerdefinierter Fehler

Diese Referenz wird aus dem Programm erzeugt und ist nur auf Englisch.

Command line: mc1sim

The options of the twin's program `mc1sim`, as its usage text gives them (86 options).

```
usage: mc1sim [options] scenario.nc ...     (batch, deterministic)
       mc1sim [--tcp PORT] [--ws PORT] [options]  (real time: TCP sender, browser UI)
       mc1sim --replay FILE [options]         (replay a recorded real-time session)
       mc1sim --play FILE.mcap [--ws PORT | --gui] (play a --mcap recording in the UI, with a timeline)
       mc1sim                                 (single-file build: the machine in its own window,
                                               a sender port on 2323 and the web view on 8765)
```

option	what it does
`--machine FILE`	machine archive (default: built in, or twin/machine/mc1.machine.json)
`--max-time S`	stop after S seconds of virtual time (default 3600 in batch, none in real time)
`--fg-quantum US`	longest foreground wait between resumes (default 50)
`--link KIND`	the sender's line: usb_cdc (the Teensy's native USB), uart (through a USB-serial adapter) or telnet (Ethernet); default the board's (machine archive controller.board)
`--baud N`	a UART at N baud (implies --link uart)
`--link-param K=V`	change a number of the link, e.g. adapter_timer_ms=1 or host_in_latency_us=0; host_dtr=0: the twin itself does not hold DTR (the native USB answers only once a host raises it), connect_dtr=0: a TCP sender does not raise it either
`--fg-cost off|L,P,S`	the foreground's CPU time in us: a line parsed, a block planned, a step segment prepared (default the board's; off: none)
`--pos X,Y,Z`	power-on position, machine mm (default from the archive)
`--sw-delay US`	limit sensors' response time (default from the archive)
`--tool-offset MM`	tool tip below the reference tool (default 0)
`--drive L0|L1|L2|L3 feed drives: L0 ideal steps, L1 closed-loop steppers moving rigid axes,`	L2 rotor, screw and table apart (coupling, torsion, Hertzian nut, play, ringing), L3 the motors' two phases and the drives' current loops on rigid axes
`--l3-dyno [V]`	the L3 motor on a dynamometer: torque against speed, as JSON (V: another supply)
`--lobes SPEC`	a side-milling cut's stability lobes by three methods (zero-order, linear time domain, semi-discretisation), JSON: "tool=flat 6 n3 L20;mat=6061-T6;ae=1;ap=1;dir=x; rpm=6000:24000:500;m=100;methods=zoa,td,sdm"
`--chatter-sim SPEC`	the cut in time, nonlinear (each tooth out of the cut where its chip is gone): the vibration and whether it chatters, JSON: "...;ap=2;rpm=15000;fz=0.02;t=0.5" (default: the machine archive's drives.model)
`--geom on|off`	the G- and T-layer: geometric errors, lead error, backlash, thermal offsets (machine archive "accuracy"; default off)
`--fidelity batch|live`	what feeds back into the simulation (docs/06 section 8.3): batch - the full resolution, as tests and the AI's practice machines have it; live - what a virtual machine at 1x keeps up with (the cut's path merged to an eighth of a cell). Default live with the window, --ws or --tcp and no --lockstep, else batch; a replay or a branch takes its recording's
`--threads N`	threads for the cut's sweep (the results are the same with any number): 1 this one alone; default the processors less one, at most 8 (or TWINWRIGHT_THREADS)
`--chatter-async on|off`	the chatter check on a thread of its own, below the simulation's: its answers taken in the order the cuts came - the same as at once, later (the summary waits for them; the views show the tool's last one meanwhile). Default on where the machine keeps the wall clock's pace (the window, --ws, --tcp; not --lockstep)
`--chatter-cache FILE|off`	the chatter check's time-domain depths kept in FILE - the same results, the second run faster. Default: the state folder's chatter-depths.bin for the window and the MCP's practice machines; off elsewhere (TWINWRIGHT_CHATTER_CACHE too)
`--set PATH=VALUE`	change a machine archive entry as it is read, e.g. drives.motor.detent_nm=0.01 or drives.alm.wired=false (repeatable)
`--mpg N`	the handwheel UART as if the firmware were built with MPG_ENABLE N (2: the handwheel's own 0x8B takes control; 0: none) - default as the archive's firmware options say (a replay: as its log says)
`--mpg-port PORT`	an external handwheel on the handwheel UART's cable, over TCP (with --mpg 2): its bytes reach the controller at the handwheel's baud rate (logged), the controller's come back
`--panel SPEC`	how the operator's buttons are wired: cycle_start=pin|sender|mpg, feed_hold=... (pin: the board's input, Pn: and its debounce; sender: '~' '!' from the sender; mpg: on the handwheel's stream) - default as the archive's controller.panel, else the sender's (a replay: as its log says)
`--fault WIRE=MODE`	wiring fault from power-on: x_min x_max ... z_switch setter alm estop estop_km1 rs485 (the VFD's bus) vfd_relay (its fault relay in the ALM chain) = open|short|intermittent (wire names follow the machine archive)

option	what it does
`--seed N`	sensors with their scatter (repeatability, response jitter) drawn from seed N; without: nominal sensors. Intermittent wires use it too
`--nvs FILE`	load EEPROM image; --save-nvs FILE writes it at exit. Batch runs start blank; real-time and replay runs from the MC-1 settings image (built in, or build/mc1.nvs made by make) unless --nvs FILE or --nvs blank
`--world DIR`	the persistent world (docs/06 8.5): the machine as the last session left it - where the axes stood, the workpiece clamped and cut, the tool, the EEPROM (written through to DIR/eeprom.bin), the temperatures and the board's clock moved on by the time it was off, the odometers; saved there at the end. The window keeps one in its state folder; batch runs and replays only with this
`--world-reset`	start the world from the factory state (its files removed first)
`--world-off S`	as if the machine had been off S seconds (else the wall clock's time since)
`--predict FILE`	with --shadow: the program the operator streams; each run of it the real machine starts is predicted by a background twin set up like the machine (homed, the workpiece, tool and offsets), flat out - its own run time against the machine's
`--nvs-through FILE`	the controller's EEPROM kept in FILE, as the board's flash keeps it: loaded from FILE if it exists (else from the image above, and FILE made), every write landing in FILE at once - settings, offsets and tools carry over to the next session. Not in replays (a recording has its start image beside it: LOG.nvs)
`--params-trace FILE the machine archive's values the run used (JSON): where each sits, its`	provenance (source, uncertainty), what it feeds, whether --set changed it
`--steps FILE`	write every step pulse (tick step_bits dir_pins)
`--spindle-trace FILE`	each millisecond the spindle turns: seconds, its angle (revolutions) and where the steps put the axes (machine mm) - a thread's pitch and phase (src/spenc.h)
`--hil-log FILE`	write what crosses between the controller and the machine as the HIL emulator's frames (src/hil.h): step groups, enables, coolant, RS-485, the machine's answers
`--hil-replay FILE`	the machine driven by such a log, the controller off (HIL's follower)
`--wav FILE`	the run's sound as the machine makes it (src/sound.h): 48 kHz stereo, 32-bit float; the same run gives the same file
`--sound-player FILE,FACTOR`	the window's player of the sound at speed FACTOR, without a device: the live frames played in this thread into a WAV (tests: the pitch at any factor)
`--sound-analyse FILE[,DBFS_AT_94[,T0,T1]]`	a WAV measured as a sound level meter reads it: LZeq, LAeq, 1/3-octave bands, the strongest lines (JSON; default -20: 94 dB SPL is -20 dBFS rms)
`--sound-listener X,Y,Z`	where --wav and --sound-player listen: model mm at the GLB pose (the machine archive's "sound"), facing the tool; default the archive's operator in front
`--sound-room off|small|workshop`	the room --wav and --sound-player hear it in (RT60 0.35 s, 0.7 s; default off: the dry sound)
`--events FILE`	write inputs, output lines and markers with ticks
`--echo`	print controller output with virtual time
`--factor F`	real-time factor (default 1)
`--www DIR`	serve the UI from DIR (default: built in, or micro_cnc/)
`--bind local|any`	listen on localhost only (the default) or on every interface (in shadow mode the sender port passes to the real machine: any lets the network move it; a twin in WSL reached from Windows needs any)
`--open`	open the web UI in the default browser
`--gui`	the native window (Windows; the default when started without arguments)
`--lang TAG`	the language (en, zh-CN, a pack's tag; default: the settings, else the system's; qps-ploc: the pseudo-language that shows untranslated texts and tight layouts)
`--lang-dir DIR`	one more folder with language packs (.po); see twin/lang/README.md
`--units U`	metric | imperial | a units.json system; --region auto|lang|de-DE; --currency EUR
`--lang-list`	the languages found; --lang-check FILE.po checks a pack; --lang-template FILE.pot writes the texts to translate
`--warp`	draw the 3D view in software (WARP) even with a graphics card
`--selftest DIR`	window self-test: off-screen, homes and runs the demo program at --factor 5, pictures and selftest.json into DIR, exit status 1 on failure
`--selftest-setup DIR`	the same for setting up workpieces (panel, dragging, keys, clamps, vise)
`--record FILE`	log real-time input for --replay
`--mcap FILE`	record signals, events, serial lines and UI frames (PlotJuggler, Foxglove)
`--vcd FILE`	record the controller's pins and the machine's lines (GTKWave, PulseView)
`--rec-period MS`	sampling period of the recordings (default 1)

option	what it does
`--stock x0,x1,y0,y1,ztop,zbot[,res]`	a block held as if bolted down (machine mm; cells default 0.02 mm)
`--setup JSON|@FILE`	the workpiece set up as on the machine: shape, size, material, fixture (clamps, vise, vblock, tape, none, fixed), position; e.g. '{"shape":"block","size":[60,40,20], "material":"6061-T6","fixture":"vise","at":[100,75]}' (see twin/core/README)
`--tool 'flat 6 n2'`	the cutter: flat|ball|bull|v|drill D [Rc|angle] [nN] (default from the archive)
`--part FILE.stl[@X,Y,Z]`	the part the program should make, its origin at X,Y,Z machine mm (default 0,0,0)
`--stock-stl FILE[,RES[,FRAME]]`	write the workpiece as a binary STL at the end (cells RES, default 0.1; FRAME machine (default), program: the program's zero, blank: its own corner)
`--stock-png FILE[,RES]`	write its height map, or the deviation from --part, as a PNG
`--branch S`	with --replay and a real-time option: replay the log to S seconds, then carry on live
`--mtconnect PORT`	MTConnect adapter (SHDR) for an agent on PORT (7878 is usual)
`--lockstep PORT`	a script drives the twin over localhost PORT (twin/py/mc1twin): virtual time moves only when it says, as fast as the host allows (api.h has the requests); 0: any free port, printed on standard output as lockstep-port N
`--api-token T`	the lockstep client's first request must be hello with token T
`--exit-with-client`	the session ends when its lockstep client goes (or none came within 30 s)
`--control PORT`	the control port (AI access, as the window's AI switch opens it): the lockstep API's requests from several local clients while time runs as usual; 0 any port
`--control-file F`	write its port and token to F as JSON (the MCP server reads it)
`--workdir DIR`	programs' versions here (default Documents/Twinwright Studio/programs; TWINWRIGHT_WORKDIR)
`--control-token T`	its token (default: a random one)
`--control-confirm yes|no|ask`	how its confirmations are answered (default yes: no window to ask)
`--own-call JSON`	{"tool":..., "args":{...}}: a tool called from the window's own interface (its own channel, the AI switch off or on), printed as "own-call N {...}" (tests; up to 12)
`--mcp`	the MCP server for AI clients on standard input and output (they start it; see twin/docs/05 section 9): background twins, and the window when its AI switch is on
`--mcp-http PORT`	the same over Streamable HTTP at http://127.0.0.1:PORT/mcp
`--ai`	(Windows) the window with its AI switch on (it remembers the switch anyway)
`--mtconnect-devices FILE`	write the matching MTConnect device model (Devices.xml; - for stdout)
`--fmi-model-description FILE`	write the FMI 2.0 modelDescription.xml of the lockstep API's FMI-shaped interface (fmi_* requests; twin/py/mc1twin/fmi.py; - for stdout)
`--shadow HOST[:PORT]|COMn|/dev/ttyACM0`	shadow mode: the sender port relays to the real machine's controller (Telnet, port 23, or its USB serial port); the twin runs the same lines alongside and compares answers, path, settings and run times (use --factor 1, the default)
`--shadow-guard`	with --shadow: each line of the sender's is looked at before it reaches the machine; a predicted crash holds it back and feed-holds the machine (the views release it)
`--version`	the program's version and the kind of build (public, internal; a public build with the test key says so)
`--licence-status [DIR] [--json] | --licence-request [DIR] | --licence-import FILE [DIR] | --licence-canon FILE | --licence-keys`	the licence files (docs/14): their state; this computer's request code for offline activation; a licence file into the folder (DIR: the machine's licence folder unless given); a file's signed bytes; the keys this build takes
`--licence-activate CODE [DIR] [--json] | --licence-trial maker|pro [DIR] [--json] | --licence-renew [DIR] [--json] | --licence-renew-due [DIR] | --licence-code CODE | --licence-server`	online (product/23, product/30): activate this computer with an activation code; ask the licence server for this computer's 30-day trial of an edition; renew the licence that is due, now; whether the window's start would renew (a day since the last try); a typed code in its canonical form; the server this build talks to (the last two send nothing)

Diese Referenz wird aus dem Programm erzeugt und ist nur auf Englisch.

Machine profile schema: MC-1

Every key of the machine archive the twin is built from (24 sections), with the value the MC-1's archive gives - for the twin's internal error model (`accuracy`, the coupling's play, and where their values come from) the keys only. The values are the model's parameters - typical values until a machine is measured - not specifications of the machine and not its accuracy; every number the twin gives about the physical world is an estimate.

`schema`

the archive format and its version

key	type	unit	value	what it is
`schema`	string		"mc1-machine/0.1"	

`generated`

what made this archive, from which inputs (hashes of params.py and wiring.py)

key	type	unit	value	what it is
`generated.by`	string		"twin/gen/gen_machine.py"	
`generated.params_sha256`	string		"ac86b6d281b81f5ca58fa7acf69c2237bee4a1a143490b3b555ad911…	
`generated.wiring_sha256`	string		"b4a817153a66f8458fe8ef3bd563b7ac0f6174733b8757570188bcf9…	
`generated.report_json`	object			the design checks' report the archive was made from: its path and when it was written
`generated.report_json.path`	string		"out/report.json"	
`generated.report_json.modified`	string		"2026-10-13T05:04:23"	

`units`

the base units

key	type	unit	value	what it is
`units.length`	string		"mm"	
`units.mass`	string		"kg"	
`units.time`	string		"s"	

`name`

key	type	unit	value	what it is
`name`	string		"MC-1"	

`coordinates`

the machine coordinate frame in words (MPos = machine - home offset)

key	type	unit	value	what it is
`coordinates`	string		"machine X 0..200 left to right, Y 0..150 tool relative t…	

`power_on_pos`

where the axes stand at power-on, machine mm (X, Y, Z)

`axes`

the three linear axes, in the order X, Y, Z

key	type	unit	value	what it is
`axes[].id`	string		"X"	
`axes[].moves`	string		"X carriage with the Z axis and spindle"	

key	type	unit	value	what it is
axes[].travel	array (2)		[0.0, 200.0]	the axis' travel [min, max], machine mm
axes[].home_dir	string		"+"	the direction homing seeks its switch
axes[].steps_per_mm	number	per mm	800.0	steps per mm (the controller's $100-$102)
axes[].dir_sign	number		-1	+1: the motor's positive direction is the axis' positive direction
axes[].direction	object			which way the motor turns the screw and the axis moves: for the drive model and the wiring's polarity
axes[].direction.motor_end	string		"+"	
axes[].direction.screw_hand	string		"right"	
axes[].direction.dir_low_turns	string		"cw (seen from the shaft end; assumed)"	
axes[].direction.machine_per_model	number		1	
axes[].direction.dir_low_moves	number		-1	
axes[].switch	object			the axis' limit / home switches: position, contact, sensor physics
axes[].switch.min	number		-2.0	
axes[].switch.max	number		202.0	
axes[].switch.contact	string		"NC"	
axes[].switch.inputs	string		"separate"	
axes[].switch.type	string		"M8 inductive NPN NC, Sn 2.0 mm"	
axes[].switch.sn_mm	number	mm	2.0	
axes[].switch.gap_design_mm	number	mm	1.0	
axes[].switch.gap_mm	array (2)	mm	[1.0, 1.0]	
axes[].switch.spot_r_cad_mm	number	mm	3.0	
axes[].switch.anchor	string		"cad"	
axes[].switch.target	array (2)		["steel", "steel"]	
axes[].switch.unit_sr	array (2)		[1.0, 1.0]	
axes[].switch.lambda_mm	number	mm	0.7	---------- limit sensors and wiring The inductive sensors' switching model follows research note F (1.3, 1.4, 7): the flag sweeps across the sensing face at gap g; the damping of the oscillator is D(g, e) = exp(-g / lambda) * F(e / w) with a logistic coverage F, and the sensor switches when it reaches the damping of a target at distance Sr straight in front: F(e_on / w) = exp(-(Sr - g) / lambda). e_on is the flag edge's position relative to the sensor axis. Sr = Sn * material factor * unit spread * (1 + tc * (T - 20 C)). The CAD places the sensors for an idealised sensing spot of radius SENSE_R (e_on = -SENSE_R): "anchor": "cad" keeps the designed switch points for the nominal sensor and moves them by what the model predicts for other gaps, materials, units and temperatures; "anchor": "model" uses the model's switch point outright (about 1.9 mm later: the design review's sensitivity case). All values unverified until measured (F 7).
axes[].switch.w_mm	number	mm	1.0	
axes[].switch.hyst_mm	number	mm	0.15	
axes[].switch.response_ms	number	ms	1.0	
axes[].switch.jitter_ms	number	ms	0.25	
axes[].switch.repeat_sigma_mm	number	mm	0.003	
axes[].switch.temp_c	number	°C	20.0	
axes[].switch.temp_coeff_per_k	number	1/K	0.0	
axes[].switch.ready_ms	number	ms	30.0	IEC 60947-5-2 "time delay before availability": after its supply comes up the output stays off (F 5: 30 ms); the controller boots later (controller.boot_ms), so it never sees it (unverified)
axes[].hard_stop	array (2)		[-8.02, 209.02]	where the moving part hits the frame [min, max], machine mm

key	type	unit	value	what it is
`axes[].moving_mass_kg`	number	kg	13.9	

`sensor_materials`

inductive sensors: switching distance reduction factor by target material

key	type	unit	value	what it is
`sensor_materials.steel`	number		1.0	
`sensor_materials.stainless`	number		0.7	
`sensor_materials.brass`	number		0.45	
`sensor_materials.aluminium`	number		0.4	
`sensor_materials.copper`	number		0.35	

`wires`

the field wiring by signal: the controller input, cable, route and length

key	type	unit	value	what it is
`wires.x_min`	object			one signal's wire: the controller input, the cable, its route and length, and the cable chains it bends in (flex)
`wires.x_min.input`	string		"X_LIMIT"	
`wires.x_min.cable`	string		"W09"	
`wires.x_min.route`	string		"固定，横梁后侧"	
`wires.x_min.length_mm`	number	mm	728.0	
`wires.x_min.flex`	string		""	
`wires.x_max`	object			
`wires.x_max.input`	string		"X_LIMIT_MAX"	
`wires.x_max.cable`	string		"W10"	
`wires.x_max.route`	string		"固定，横梁后侧"	
`wires.x_max.length_mm`	number	mm	488.0	
`wires.x_max.flex`	string		""	
`wires.y_min`	object			
`wires.y_min.input`	string		"Y_LIMIT"	
`wires.y_min.cable`	string		"W12"	
`wires.y_min.route`	string		"固定，底板下方"	
`wires.y_min.length_mm`	number	mm	167.0	
`wires.y_min.flex`	string		""	
`wires.y_max`	object			
`wires.y_max.input`	string		"Y_LIMIT_MAX"	
`wires.y_max.cable`	string		"W11"	
`wires.y_max.route`	string		"固定，底板下方"	
`wires.y_max.length_mm`	number	mm	212.0	
`wires.y_max.flex`	string		""	
`wires.z_switch`	object			
`wires.z_switch.input`	string		"Z_LIMIT"	
`wires.z_switch.cable`	string		"W08"	
`wires.z_switch.route`	string		"Z 随动线束 + X 拖链，与主轴线分开绑扎"	
`wires.z_switch.length_mm`	number	mm	1493.0	
`wires.z_switch.flex`	string		"XZ"	
`wires.setter`	object			
`wires.setter.input`	string		"PROBE"	

key	type	unit	value	what it is
wires.setter.cable	string		"W13"	
wires.setter.route	string		"Y 拖链"	
wires.setter.length_mm	number	mm	1267.0	
wires.setter.flex	string		"Y"	
wires.alm	object			
wires.alm.input	string		"MOTOR_FAULT"	
wires.alm.cable	string		""	
wires.alm.route	string		"电控箱内：三台驱动器 ALM 与变频器故障继电器串联"	
wires.alm.length_mm	number	mm	0.0	
wires.alm.flex	string		""	
wires.estop	object			
wires.estop.input	string		"E-STOP"	
wires.estop.cable	string		""	
wires.estop.route	string		"电控箱：急停 SB0 第二组常闭（SB0-2）"	
wires.estop.length_mm	number	mm	0.0	
wires.estop.flex	string		""	
wires.estop_km1	object			
wires.estop_km1.input	string		""	
wires.estop_km1.cable	string		""	
wires.estop_km1.route	string		"电控箱：急停 SB0 第一组常闭（SB0-1），在 KM1 线圈的自锁回路里"	
wires.estop_km1.length_mm	number	mm	0.0	
wires.estop_km1.flex	string		""	
wires.rs485	object			
wires.rs485.input	string		"RS485"	
wires.rs485.cable	string		""	
wires.rs485.route	string		"电控箱内：控制板 RS485 A/B → 变频器 RS+/RS-，屏蔽双绞线"	
wires.rs485.length_mm	number	mm	0.0	
wires.rs485.flex	string		""	
wires.vfd_relay	object			
wires.vfd_relay.input	string		"MOTOR_FAULT"	
wires.vfd_relay.cable	string		""	
wires.vfd_relay.route	string		"电控箱内：变频器故障继电器的常闭触点 TB–TC，串在 ALM 链里（发现 107）"	
wires.vfd_relay.length_mm	number	mm	0.0	
wires.vfd_relay.flex	string		""	

faults

wiring fault models (glitches, intermittent contacts)

key	type	unit	value	what it is
faults.glitch_us	number	µs	50.0	injected faults: a glitch opens the contact this long; an intermittent wire drops out while its cable flexes (per metre of travel of the axes it follows), for a random time
faults.intermittent	object			an intermittent wire: it drops out while its cable flexes, so often per metre of travel of the axes it follows, each time for a random time in the range (ms)
faults.intermittent.per_m	number	per m	5.0	
faults.intermittent.ms	array (2)		[0.2, 3.0]	

`drives`

the feed drives: ball screws, closed-loop steppers, their power (KM1), the L1/L2 models

key	type	unit	value	what it is
`drives.model`	string		"L1"	
`drives.note`	string		"rigid axis driven by a closed-loop stepper: magnetic spr…	
`drives.lead_mm`	number	mm	4.0	
`drives.motor`	object			the stepper motor: holding torque, rotor inertia, steps, detent torque, phase resistance and inductance, torque constant, rated current, iron loss
`drives.motor.type`	string		"NEMA23 closed-loop hybrid stepper, 23HS22-2804D-E1000 cl…	
`drives.motor.holding_nm`	number	N·m	1.26	
`drives.motor.rotor_inertia_kgm2`	number	kg·m²	3e-05	
`drives.motor.full_steps_per_rev`	number	per revolution	200	
`drives.motor.detent_nm`	number	N·m	0.05	
`drives.motor.power_up_jump`	boolean		true	the drive's first current after a power-up (its phase reset): the rotor snaps to the nearest stable position of that state, up to two full steps either way (architecture 4.4; unverified on the drive)
`drives.motor.phase_resistance_ohm`	number	Ω	0.9	L3 (research note D, L3): the phases - R and L of the 2.8 A class (typical catalogue values, unverified), the torque constant fitted to the 48 V pull-out plateau (0.8 N m at 2.8 A; the 1.26 N m holding torque is both phases at rated current), the detent's amplitude as above
`drives.motor.phase_inductance_mh`	number	mH	2.9	
`drives.motor.torque_constant_nm_a`	number	N·m/A	0.286	
`drives.motor.rated_current_a`	number	A	2.8	
`drives.motor.iron_loss_nm_per_krpm`	number	N·m per 1000 rpm	0.0377	the laminations' hysteresis and eddy losses as a drag growing with speed: fitted so that L3 on its dynamometer gives the 48 V pull-out curve above (9 W at 1500 rpm; to be measured, architecture 7.3 step 8)
`drives.coupling_inertia_kgm2`	number	kg·m²	2.6e-06	
`drives.pullout_48v_rpm_nm`	array (9)	rpm/(N·m)		
`drives.drive`	object			the closed-loop drive: encoder counts, holding current, damping, the following error it alarms at and how that alarm is reset, current loop, DC bus, the L3 sub-steps
`drives.drive.type`	string		"CL57T class closed-loop drive, 48 V"	
`drives.drive.loop`	string		"closed"	
`drives.drive.encoder_counts_per_rev`	number	per revolution	4000	
`drives.drive.holding_current`	number		0.5	
`drives.drive.damping_ratio`	number		0.5	
`drives.drive.smoothing_ms`	number	ms	0.0	
`drives.drive.following_error_limit_counts`	number		400	
`drives.drive.alarm_reset`	string		"power"	
`drives.drive.open_loop_below_rpm`	number	rpm	0	
`drives.drive.current_loop_hz`	number	Hz	2000.0	then only the holding current at standstill (0: always closed) L3: its current loop's bandwidth, its bus capacitance and overvoltage alarm (assumed), 5 us steps
`drives.drive.bus_capacitance_uf`	number	µF	470.0	
`drives.drive.overvoltage_v`	number	V	60.0	
`drives.drive.l3_sub_steps`	number		20	

key	type	unit	value	what it is
drives.drive.max_phase_advance_deg	number	°	0.0	
drives.alm	object			the alarm chain: the drives' ALM outputs and the spindle drive's fault relay in series to the controller's motor-fault input
drives.alm.wired	boolean		true	
drives.alm.vfd_relay	boolean		true	
drives.alm.chain	string		"the three ALM outputs and the VFD's fault relay (its mul…	
drives.power	object			the drives' supply: the 48 V supply behind the contactor KM1, its timing, the supply's hold-up and rise, the drives' boot
drives.power.source	string		"PSU1 48 V behind KM1 (drives, VFD); the 24 V supply befo…	
drives.power.km1	string		"24 V DC coil, latched by the START button SB1 through it…	
drives.power.km1_latched	boolean		true	
drives.power.km1_drop_ms	number	ms	15	
drives.power.km1_pickup_ms	number	ms	25	
drives.power.psu_holdup_ms	number	ms	40	
drives.power.psu_rise_ms	number	ms	150	
drives.power.drive_boot_ms	number	ms	800	
drives.step_s	number	s	0.0001	
drives.l2	object			L2 (--drive L2): the axial chain between the motor side and the table - stiffness by the nut's distance from its BK bearing (the design's hand calculation at 100 N), the nut's play (the accuracy section's backlash), modal damping assumed; integrated in sub-steps of step_s. The drive's current loop (lag) and velocity loop (damping of the motor side on its magnetic spring) are assumed: a closed-loop drive is tuned not to overshoot - fit both to encoder logs (architecture 7.3 step 1) The nut's play closes with an impact: the grease film makes it strongly damped at a few mm/s (Hunt-Crossley damping, growing with the penetration; restitution assumed)
drives.l2.zeta_axial	number		0.03	
drives.l2.sub_steps	number		5	
drives.l2.current_lag_ms	number	ms	0.2	
drives.l2.servo_damping_ratio	number		0.7	
drives.l2.smoothing_ms	number	ms	1.0	
drives.l2.contact_restitution	number		0.0	
drives.l2.impact_speed_mm_s	number	mm/s	5.0	
drives.l2.masses	number		3	three masses (note D, L2): the coupling's mode sits near 1.75 kHz - 20 sub-steps of 5 us; its damping assumed; the friction model of guides and nut: "karnopp" (Stribeck + viscous with a stick band) or "lugre" (pre-sliding, friction lag at reversals)
drives.l2.sub_steps_3	number		20	
drives.l2.zeta_coupling	number		0.05	
drives.l2.friction	string		"karnopp"	
drives.l2.stribeck_ratio	number		1.25	
drives.l2.stribeck_v_mm_s	number	mm/s	2.0	
drives.l2.viscous_n_s_mm	number	N·s/mm	0.02	
drives.l2.lugre_sigma0_n_um	number	N/µm	10.0	
drives.axes	array (3)			per axis (X, Y, Z): screw, inertias, masses, friction, the L2 elastic model
drives.axes[].id	string		"X"	
drives.axes[].screw_length_mm	number	mm	319.0	

key	type	unit	value	what it is
drives.axes[].screw_inertia_kgm2	number	kg·m²	3.2918581767123845e-06	
drives.axes[].rotating_inertia_kgm2	number	kg·m²	3.589185817671239e-05	
drives.axes[].moving_mass_kg	number	kg	13.9	
drives.axes[].equivalent_mass_kg	number	kg	102.45961035603891	
drives.axes[].gravity_n	number	N	0.0	
drives.axes[].friction_n	number	N	91.62684499333461	
drives.axes[].l2	object			the axis in the L2 drive model (--drive L2): the masses, the axial and torsional stiffness by the nut's distance from the fixed bearing, friction, the coupling, the frame under the bearing
drives.axes[].l2.rotating_mass_kg	number	kg	88.5596103560389	
drives.axes[].l2.table_mass_kg	number	kg	13.9	
drives.axes[].l2.k_axial_n_um	array (2)	N/µm		
drives.axes[].l2.friction_motor_n	number	N	81.68140899333461	
drives.axes[].l2.friction_table_n	number	N	9.945436	
drives.axes[].l2.motor_mass_kg	number	kg	77.22965443852422	three masses: the rotor with half the coupling, the screw with the other half, the table
drives.axes[].l2.screw_mass_kg	number	kg	11.329955917514674	
drives.axes[].l2.coupling_n_um	number	N/µm	3817.0695021213087	
drives.axes[].l2.torsion_gj_nmm2	number	N·mm²	64770810.92360646	the screw's torsion from the coupling to the nut: its root section's G J over the BK journal, the drive journal (as lengths of root section) and the distance BK -> nut
drives.axes[].l2.torsion_extra_mm	number	mm	53.184712792968746	
drives.axes[].l2.k_rest_n_um	array (2)	N/µm		the axial chain without the nut (shaft, BK, mount), by the nut's distance from the BK
drives.axes[].l2.nut_hertz_n_um15	number	N/µm^1.5	79.28988773038022	
drives.axes[].l2.frame	object			the frame under the axis' fixed bearing as one mode along the axis: stiffness, mass, damping
drives.axes[].l2.frame.k_n_um	number	N/µm	33.2	
drives.axes[].l2.frame.m_kg	number	kg	17.65	
drives.axes[].l2.frame.zeta	number		0.02	

drives.axes[].l2.coupling_play_um

The coupling's play, part of the twin's internal error model like the backlash: its value is not published.

key	type	unit	what it is
drives.axes[].l2.coupling_play_um	number	µm	

accuracy

geometric and thermal errors of the tool relative to the work (Phase 4)

The twin's internal error-model parameters, off by default; their values are not published.

key	type	unit	what it is
accuracy.enabled	boolean		
accuracy.note	string		
accuracy.units	string		
accuracy.sign	string		
accuracy.axes	object		
accuracy.axes.X	object		
accuracy.axes.X.lead_um	array (45)	µm	
accuracy.axes.X.periodic_um	number	µm	

key	type	unit	what it is
accuracy.axes.X.periodic_mm	number	mm	
accuracy.axes.X.periodic_phase	number		
accuracy.axes.X.straight_um	object	µm	
accuracy.axes.X.straight_um.Y	array (45)		
accuracy.axes.X.straight_um.Z	array (45)		
accuracy.axes.X.angle_urad	object	µrad	
accuracy.axes.X.angle_urad.A	array (45)		
accuracy.axes.X.angle_urad.B	array (45)		
accuracy.axes.X.angle_urad.C	array (45)		
accuracy.axes.X.backlash_um	number	µm	
accuracy.axes.X.gravity_loaded	boolean		
accuracy.axes.Y	object		
accuracy.axes.Y.lead_um	array (35)	µm	
accuracy.axes.Y.periodic_um	number	µm	
accuracy.axes.Y.periodic_mm	number	mm	
accuracy.axes.Y.periodic_phase	number		
accuracy.axes.Y.straight_um	object	µm	
accuracy.axes.Y.straight_um.X	array (35)		
accuracy.axes.Y.straight_um.Z	array (35)		
accuracy.axes.Y.angle_urad	object	µrad	
accuracy.axes.Y.angle_urad.A	array (35)		
accuracy.axes.Y.angle_urad.B	array (35)		
accuracy.axes.Y.angle_urad.C	array (35)		
accuracy.axes.Y.backlash_um	number	µm	
accuracy.axes.Y.gravity_loaded	boolean		
accuracy.axes.Z	object		
accuracy.axes.Z.lead_um	array (21)	µm	
accuracy.axes.Z.periodic_um	number	µm	
accuracy.axes.Z.periodic_mm	number	mm	
accuracy.axes.Z.periodic_phase	number		
accuracy.axes.Z.straight_um	object	µm	
accuracy.axes.Z.straight_um.X	array (21)		
accuracy.axes.Z.straight_um.Y	array (21)		
accuracy.axes.Z.angle_urad	object	µrad	
accuracy.axes.Z.angle_urad.A	array (21)		
accuracy.axes.Z.angle_urad.B	array (21)		
accuracy.axes.Z.angle_urad.C	array (21)		

key	type	unit	what it is
accuracy.axes.Z.backlash_um	number	µm	
accuracy.axes.Z.gravity_loaded	boolean		
accuracy.square_urad	object	µrad	
accuracy.square_urad.xy	number		
accuracy.square_urad.xz	number		
accuracy.square_urad.yz	number		
accuracy.abbe	object		Abbe arms: where each axis' angular errors are referred to (params.py geometry, model frame: y towards the beam, z up from the base top; the reference tool tip is at z = tip_z0 + Z)
accuracy.abbe.tip_z0	number		
accuracy.abbe.X	object		
accuracy.abbe.X.y	number		
accuracy.abbe.X.z	number		
accuracy.abbe.Z	object		
accuracy.abbe.Z.y	number		
accuracy.abbe.Z.dz	number		
accuracy.abbe.Y	object		
accuracy.abbe.Y.x	number		
accuracy.abbe.Y.y	number		
accuracy.abbe.Y.z	number		
accuracy.thermal	object		
accuracy.thermal.alpha_um_m_k	number	µm/(m·K)	
accuracy.thermal.screw_dt_k	array (3)	K	
accuracy.thermal.bk_at	array (3)		
accuracy.thermal.spindle_growth_um	number	µm	
accuracy.thermal.dynamic	object		first-order dynamics (research note D 8 T-layer, "second version"): each screw warmed by its nut and BK drag times its speed through a thermal resistance, the spindle grown by its bearings (by speed) and its motor's copper (by current) - time constants and gains assumed, to be fitted to a 2 h warm-up log (architecture 7.3 step 14); off unless asked for (the static offsets above, and %thermal, stay as set)
accuracy.thermal.dynamic.enabled	boolean		
accuracy.thermal.dynamic.screw_k_per_w	number	K/W	
accuracy.thermal.dynamic.screw_tau_s	number	s	
accuracy.thermal.dynamic.spindle_um_per_w	number	µm/W	
accuracy.thermal.dynamic.spindle_tau_s	number	s	
accuracy.thermal.dynamic.spindle_bearing_w_at_24000	number	W	
accuracy.thermal.dynamic.spindle_copper_w_rated	number	W	
accuracy.thermal.frame	object		
accuracy.thermal.frame.alpha_um_m_k	number	µm/(m·K)	
accuracy.thermal.frame.dt_k	number	K	
accuracy.thermal.frame.tool_to_table_um_k	number	µm/K	
accuracy.thermal.frame.capacity_j_k	number	J/K	
accuracy.thermal.frame.g_w_k	number	W/K	
accuracy.thermal.frame.motors_w	number	W	
accuracy.thermal.frame.spindle_share	number		
accuracy.thermal.ambient_dt_k	number	K	the room above the reference (20 C): the frame follows it (at once without the dynamics)
accuracy.thermal.part	object		the workpiece (research note L 8): 10-25 % of the cutting power goes into it (0.15 taken), out through the fixture - 0.2 K/W with good contact (vise, clamps), 1 K/W on tape or lying loose
accuracy.thermal.part.heat_share	number		

key	type	unit	what it is
`accuracy.thermal.part.g_good_w_k`	number	W/K	
`accuracy.thermal.part.g_poor_w_k`	number	W/K	
`accuracy.source`	string		

`spindle`

the spindle drive: the Huanyang VFD (HY protocol, PD parameters), the motor, the pump

key	type	unit	value	what it is
`spindle.vfd`	object			the spindle drive (Huanyang HY series): rating, supply, the RS-485 port and the HY protocol, timing, the DC bus and its protection levels, its parameters
`spindle.vfd.model`	string		"Huanyang HY01D523B (HY series)"	
`spindle.vfd.rated_power_w`	number	W	1500	
`spindle.vfd.rated_current_a`	number	A	7.0	
`spindle.vfd.supply`	string		"1 x 220 V through KM1 (build.py ELECTRICAL)"	
`spindle.vfd.port`	string		"RS-485 to RS+/RS-/DCM (wiring.py PIN_MAP 主轴)"	
`spindle.vfd.protocol`	string		"HY: address, function, length, data, CRC-16 (Huanyang v1)"	
`spindle.vfd.reply_latency_ms`	number	ms	5.0	
`spindle.vfd.min_gap_ms`	number	ms	5.0	
`spindle.vfd.boot_s`	number	s	1.5	
`spindle.vfd.bus_capacitance_uf`	number	µF	680.0	
`spindle.vfd.lu_v`	number	V	200.0	DC bus: without mains it feeds the control and the motor until undervoltage (LU: output off), then the control's own supply stops below control_off_v
`spindle.vfd.control_off_v`	number	V	100.0	
`spindle.vfd.control_w`	number	W	15.0	
`spindle.vfd.oc_factor`	number		2.0	
`spindle.vfd.overload_factor`	number		1.5	
`spindle.vfd.overload_s`	number	s	60.0	
`spindle.vfd.dc_bus_v`	number	V	311.0	
`spindle.vfd.factory`	object			the drive's PD parameters as they leave the factory (its manual)
`spindle.vfd.factory.PD000`	number		0	
`spindle.vfd.factory.PD001`	number		0	
`spindle.vfd.factory.PD002`	number		0	
`spindle.vfd.factory.PD004`	number		50.0	
`spindle.vfd.factory.PD005`	number		50.0	
`spindle.vfd.factory.PD006`	number		2.5	
`spindle.vfd.factory.PD007`	number		0.5	
`spindle.vfd.factory.PD008`	number		220.0	
`spindle.vfd.factory.PD009`	number		15.0	
`spindle.vfd.factory.PD010`	number		8.0	
`spindle.vfd.factory.PD011`	number		0.0	
`spindle.vfd.factory.PD014`	number		10.0	
`spindle.vfd.factory.PD015`	number		10.0	
`spindle.vfd.factory.PD026`	number		0	

key	type	unit	value	what it is
`spindle.vfd.factory.PD119`	number		150	
`spindle.vfd.factory.PD141`	number		220.0	
`spindle.vfd.factory.PD142`	number		7.0	
`spindle.vfd.factory.PD143`	number		4	
`spindle.vfd.factory.PD144`	number		1440	
`spindle.vfd.factory.PD163`	number		0	
`spindle.vfd.factory.PD164`	number		1	
`spindle.vfd.factory.PD165`	number		0	
`spindle.vfd.factory.PD118`	number		1	the protection and compensation group as the twin reads it (numbers and meanings from the manual's list, unverified on the unit): PD118 overvoltage stall prevention (1 on: the decel ramp waits while the DC bus is high), PD120 stall prevention level at constant speed (% of PD142), PD121 its deceleration time (s), PD123 over-torque detection (0 off, 1 warn at constant speed, 2 trip), PD124 its level (% of PD142), PD125 its time (s), PD147 slip compensation (% of the rated slip made up at rated current; 0 off)
`spindle.vfd.factory.PD120`	number		150	
`spindle.vfd.factory.PD121`	number		5.0	
`spindle.vfd.factory.PD123`	number		0	
`spindle.vfd.factory.PD124`	number		150	
`spindle.vfd.factory.PD125`	number		0.1	
`spindle.vfd.factory.PD147`	number		0	
`spindle.vfd.factory.PD041`	number		8	PD041 the PWM carrier: 0-15 for 0.7, 1, 1.5, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 15, 17, 20 kHz; the manual's factory value for the A01D523B (the HY01D523B) is 8 = 8 kHz (research note N 2.3). Only the sound uses it
`spindle.vfd.params`	object			the PD parameters commissioning sets (the design's table wiring.VFD_PARAMS)
`spindle.vfd.params.PD001`	number		2	
`spindle.vfd.params.PD002`	number		2	
`spindle.vfd.params.PD004`	number		400.0	
`spindle.vfd.params.PD005`	number		400.0	
`spindle.vfd.params.PD006`	number		20.0	
`spindle.vfd.params.PD007`	number		0.5	
`spindle.vfd.params.PD008`	number		220.0	
`spindle.vfd.params.PD009`	number		16.0	
`spindle.vfd.params.PD010`	number		4.0	
`spindle.vfd.params.PD011`	number		100.0	
`spindle.vfd.params.PD014`	number		6.0	
`spindle.vfd.params.PD015`	number		6.0	
`spindle.vfd.params.PD141`	number		220.0	
`spindle.vfd.params.PD142`	number		5.0	
`spindle.vfd.params.PD143`	number		2	
`spindle.vfd.params.PD144`	number		3000	
`spindle.vfd.params.PD163`	number		1	
`spindle.vfd.params.PD164`	number		2	
`spindle.vfd.params.PD165`	number		3	
`spindle.motor`	object			the spindle motor, an induction motor: rating, poles, slip, breakdown torque, no-load current, inertia, friction
`spindle.motor.model`	string		"GDZ65-800"	
`spindle.motor.rated_power_w`	number	W	800.0	
`spindle.motor.rated_voltage_v`	number	V	220.0	
`spindle.motor.rated_current_a`	number	A	5.0	
`spindle.motor.rated_freq_hz`	number	Hz	400.0	

key	type	unit	value	what it is
spindle.motor.poles	number		2	
spindle.motor.rated_rpm	number	rpm	24000.0	
spindle.motor.rated_slip	number		0.04	
spindle.motor.breakdown_torque_ratio	number		2.0	
spindle.motor.no_load_current_a	number	A	2.0	
spindle.motor.inertia_kgm2	number	kg·m²	8e-05	
spindle.motor.friction_nm	number	N·m	0.005	
spindle.motor.viscous_nms	number	N·m·s	1e-05	
spindle.coolant_pump	string		"on relay K2 (COOLANT flood), supplied through KM1 (wirin…	
spindle.encoder	object			an incremental encoder on the spindle (G33, G76): lines a revolution and an index pulse once a revolution
spindle.encoder.ppr	number		0	lines a revolution (0: none); the controller's $38 is this (the other firmware: one channel's pulses) or 4 times it (the kernel: quadrature counts)

supply

the 24 V supply of controller, sensors and relays

key	type	unit	value	what it is
supply.psu24	string		"PSU2 24 V 60 W, before KM1: controller, sensors, the coi…	QF1 (the utility) feeds PSU2 directly and, through KM1, PSU1 and the VFD (wiring.py)
supply.psu24_holdup_ms	number	ms	80.0	
supply.psu24_rise_ms	number	ms	100.0	

tool_setter

the tool length sensor: where it stands (machine X, Y), its top (machine Z)

key	type	unit	value	what it is
tool_setter.x	number		25.0	
tool_setter.y	number		150.0	
tool_setter.top_z	number		-53.0	the contact (research note F 7, all unverified): it opens after the plunger has gone down pretravel_mm, the point scatters by repeat_sigma_mm (with --seed), it closes again reset_hyst_mm higher, and it bounces (closes for a moment) bounces times within bounce_ms as it opens (mech-r22: it sits SETTER_RECESS deep in a pocket in the table top - its top SETTER_TOP over the table)
tool_setter.overtravel	number		3.0	
tool_setter.contact	string		"NC"	
tool_setter.diameter	number		30.0	
tool_setter.pretravel_mm	number	mm	0.2	
tool_setter.repeat_sigma_mm	number	mm	0.001	
tool_setter.reset_hyst_mm	number	mm	0.01	
tool_setter.bounces	number		2	
tool_setter.bounce_ms	number	ms	0.5	

touch_probe

a touch probe tool (ball stylus on the PROBE input): ideal - it switches as the ball touches - or typical: a kinematic seat that opens once the stylus has given way by its pretravel (by the approach, scattered with a seed), closes again less deflected; its stylus breaks past its travel (docs/06 12.31)

key	type	unit	value	what it is
touch_probe.model	string		"ideal"	

key	type	unit	value	what it is
touch_probe.stylus_travel_mm	number	mm	2.0	A touch probe is a tool ("probe 2 L30": a ball stylus on the PROBE input with the setter). "ideal": it switches as the ball touches; "typical": a kinematic touch-trigger probe's seat opens once the stylus has given way by its pretravel - along Z, in XY with three lobes round the stylus (three seats), each touch's scattered by repeatability_sigma_mm with --seed - and closes again release_hysteresis_mm less deflected (docs/06 12.31). The MC-1's probe is an option not chosen yet: the typical values are orders of magnitude for a hobby-class probe, an assumption (research O: such probes claim 1-5 um repeatability, unverified), no specification of any probe.
touch_probe.geometry	object			the probe's shape in the collision zone (the jaws, the clamps, the workpiece): its stylus over the ball, its body from body_from_mm above the ball's lowest point up to the collet; 0: not given
touch_probe.geometry.stylus_d_mm	number	mm	1.5	the stylus' diameter over the ball (thinner than the ball: the ball stands for it)
touch_probe.geometry.body_d_mm	number	mm	20.0	the body's diameter (0: no body)
touch_probe.geometry.body_from_mm	number	mm	15.0	where the body starts, above the ball's lowest point (the stylus' length below it)
touch_probe.styli	array (3)			the styli by ball size (1.1, e09's Ø2 H7 features: a Ø2 ball does not fit) - each its own stem and where the body starts above the ball's lowest point; a probe tool takes the nearest ball's (within 25 %), else geometry's. Typical M2 ruby styli, an assumption until the user's are entered
touch_probe.styli[].ball_d_mm	number	mm	2.0	
touch_probe.styli[].stylus_d_mm	number	mm	1.5	
touch_probe.styli[].body_from_mm	number	mm	15.0	
touch_probe.typical	object			the typical probe's model: pretravel along Z and sideways, the three-lobed pattern, repeatability, release hysteresis - typical of touch-trigger probes, until a probe is calibrated
touch_probe.typical.pretravel_z_mm	number	mm	0.005	
touch_probe.typical.pretravel_xy_mm	number	mm	0.015	
touch_probe.typical.lobing_xy_mm	number	mm	0.005	
touch_probe.typical.lobing_count	number		3	
touch_probe.typical.lobing_phase_deg	number	°	0.0	
touch_probe.typical.repeatability_sigma_mm	number	mm	0.001	
touch_probe.typical.release_hysteresis_mm	number	mm	0.003	

sound

where the sounds come from (model mm at the GLB pose, the motion group each rides on) and the default listener, for the sound engine's panning

key	type	unit	value	what it is
sound.frame	string		"model mm, Z up, at the GLB pose (viewer.glb_pose); moves...	
sound.sources	object			the machine's sources of sound, by name
sound.sources.x_motor	object			one source of sound: where it is (model mm at the GLB pose) and the motion group it rides on
sound.sources.x_motor.at	array (3)		[254.0, 160.0, 250.0]	
sound.sources.x_motor.moves	string		"frame"	
sound.sources.y_motor	object			
sound.sources.y_motor.at	array (3)		[0.0, 245.0, 22.0]	
sound.sources.y_motor.moves	string		"frame"	
sound.sources.z_motor	object			
sound.sources.z_motor.at	array (3)		[0.0, 91.0, 467.0]	
sound.sources.z_motor.moves	string		"X"	
sound.sources.spindle	object			
sound.sources.spindle.at	array (3)		[0.0, 0.0, 206.5]	
sound.sources.spindle.moves	string		"Z"	
sound.sources.tool_tip	object			

key	type	unit	value	what it is
sound.sources.tool_tip.at	array (3)		[0.0, 0.0, 59.0]	
sound.sources.tool_tip.moves	string		"Z"	
sound.sources.box	object			
sound.sources.box.at	array (3)		[430.0, 50.0, 140.0]	
sound.sources.box.moves	string		"frame"	
sound.sources.setter	object			
sound.sources.setter.at	array (3)		[-75.0, 75.0, 86.0]	
sound.sources.setter.moves	string		"Y"	
sound.sources.pump	object			
sound.sources.pump.at	array (3)		[-335.0, 0.0, 40.0]	
sound.sources.pump.moves	string		"frame"	
sound.listener	object			where the listener stands and what it faces (model mm)
sound.listener.at	array (3)		[0.0, -695.0, 559.0]	
sound.listener.looks_at	array (3)		[0.0, 0.0, 99.0]	
sound.radiation	object			the cut's sound through the machine's modes (cutting.dynamics): Pa/N well above them, each mode's share of it (the rest radiates flat)
sound.radiation.pa_per_n	number	Pa/N	0.005	the cut's sound: the force's harmonics through the machine's modes (cutting.dynamics: the structure's X and Y modes, the tool's) as their accelerations radiate (research note N 2.4) - pa_per_n well above the modes, each mode's share of it (a peak of share / (2 zeta) at its frequency), the shares' rest flat
sound.radiation.structure_x	number		0.35	
sound.radiation.structure_y	number		0.35	
sound.radiation.tool	number		0.3	

collision

collision proxies: the table, the spindle stack above the tool tip (coaxial cylinders), what stands on the table (table-local mm, origin at its centre)

key	type	unit	value	what it is
collision.table	object			the table's proxy: its size, its origin in machine XY, the reference tool tip's machine Z on it
collision.table.size	array (2)		[300.0, 220.0]	
collision.table.origin_machine_xy	array (2)		[100.0, 75.0]	
collision.table.ref_tip_on_table_z	number		-80.0	
collision.spindle	array (6)			
collision.spindle[].id	string		"tool"	
collision.spindle[].r	number		3.0	
collision.spindle[].h	array (2)		[0.0, 15.5]	
collision.table_items	array (7)			
collision.table_items[].id	string		"setter_flange"	
collision.table_items[].shape	string		"box"	
collision.table_items[].min	array (3)		[-107.0, 66.0, -11.0]	
collision.table_items[].max	array (3)		[-43.0, 84.0, -6.0]	
collision.groups	array (3)			
collision.groups[].id	string		"z_slide"	
collision.groups[].moves	string		"XZ"	
collision.groups[].min	array (3)		[-73.5, 45.0, 95.0]	
collision.groups[].max	array (3)		[73.5, 65.0, 255.0]	

key	type	unit	value	what it is
`collision.near_miss_mm`	number	mm	1.0	a gap this small without contact is a near miss (0: none)

`cutting`

cutting-force coefficients (Altintas' linear edge-force model), materials, tool compliance and structural dynamics for the stock model

key	type	unit	value	what it is
`cutting.material`	string		"6061-T6 (assumed, research note C 1.5)"	
`cutting.ktc_n_mm2`	number	N/mm²	750.0	
`cutting.krc_n_mm2`	number	N/mm²	250.0	
`cutting.kac_n_mm2`	number	N/mm²	150.0	
`cutting.kte_n_mm`	number	N/mm	5.0	
`cutting.kre_n_mm`	number	N/mm	5.0	
`cutting.kae_n_mm`	number	N/mm	2.0	
`cutting.kc_plunge_j_mm3`	number	J/mm³	2.4	
`cutting.tool_default`	string		"flat 6 n2"	
`cutting.stock_res_mm`	number	mm	0.02	
`cutting.chunk_mm`	number	mm	0.5	the cut's time resolution (A10): chunks of 0.5 mm / 20 ms, or "tooth": a tooth period (at most 0.05 mm) at a time; polyline points within merge_tol_um of a segment's chord merged into it (the drives' tiny deviations at L1/L2 stay in the swept surface); the height pyramid skips the blocks the tool passes over; the sweep looks only at the cells each segment can reach below the blank's top (its profile's inverse: a V-bit engraving shallow reaches a tenth of a millimetre - round 16; false: as before, the same cut)
`cutting.chunk_ms`	number	ms	20.0	
`cutting.force_step`	string		"chunk"	
`cutting.merge_tol_um`	number	µm	0.1	
`cutting.pyramid`	boolean		true	
`cutting.sweep_reach`	boolean		true	
`cutting.variation`	boolean		true	the shear coefficients follow chip thickness and cutting speed where a material has orthogonal cutting data ("oblique"); false: constant coefficients everywhere (--set cutting.variation=0)
`cutting.wear`	object			the tools' wear book: kept and reported only (the cut stays a sharp tool's); the wear band's curve, the edge's clearance angle, a small tool's criterion
`cutting.wear.track`	boolean		true	the tools' wear book (1.1, twin/core/src/wear.h; handover/virtual-cnc-tools/wear_design.md): track - kept and reported only, the cut a sharp tool's; the wear band's curve (run in to run_in[1] VB_c by u = run_in[0], late_slope times as steep past its life), the edge's clearance angle (VB tan alpha: how far it goes back); a small tool's criterion at most vb_c_d x D, not under vb_c_min_mm (micro-milling studies: VB 12 um ends a 0.5 mm tool's life, 37.5 um a 1 mm one's - about 3 % of D; wear_life_data.md section 2)
`cutting.wear.alpha_deg`	number	°	10.0	
`cutting.wear.run_in`	array (2)		[0.05, 0.3]	
`cutting.wear.late_slope`	number		3.0	
`cutting.wear.vb_c_d`	number		0.03	
`cutting.wear.vb_c_min_mm`	number	mm	0.01	
`cutting.tool_geometry`	object			the end mills' geometry for that transformation: a two-flute aluminium cutter (the values Ozsoy used); the helix also spreads the force's harmonics (stock.c force_harmonics, docs/06 M3)

key	type	unit	value	what it is
cutting.tool_geometry.helix_deg	number	°	30.0	
cutting.tool_geometry.rake_deg	number	°	10.5	
cutting.tool_strength	object			the end mill's strength (research note L 4.4; the twin's M4): carbide rod's transverse rupture strength, the share of it a tool takes at once, the flute core's diameter for strength (x D)
cutting.tool_strength.trs_mpa	number	MPa	4000.0	
cutting.tool_strength.allowable	number		0.35	
cutting.tool_strength.core_d	number		0.7	
cutting.force_model	string		"tooth"	the cutting forces (V0 P1, stock.c and tooth.c): "tooth" - the tooth-level kernel, disc by disc and tooth by tooth (engagement from the cells cut, the tool's profile, the helix, runout, the minimum chip, Kienzle's law where a material has it); "chunk" - one engagement arc per chunk and the flat end mill's closed forms, as before. The runout at the tool's tip (TIR; research note L 3.3: ER11 collet and spindle 5-15 um), the offset's direction drawn per cutter from the seed unless phase_deg gives it; envelope: the outermost tooth cuts the part, the cutter reaching R + rho max cos(lambda - j 2 pi / N) - holes larger, bosses smaller (finding 187; false: the nominal radius, as before); the edge of a new carbide tool (radius; the minimum chip as a share of it, aluminium 0.2-0.4: note L 3.2)
cutting.runout	object			the cutter's runout in the collet (TIR), its direction (null: drawn for each tool from the session's seed), and whether the cut takes its envelope (finding 187)
cutting.runout.tir_um	number	µm	10.0	
cutting.runout.phase_deg	null	°	null	
cutting.runout.envelope	boolean		true	
cutting.edge	object			the cutting edge: its radius, and the minimum chip as a share of it - a thinner chip only ploughs
cutting.edge.radius_um	number	µm	2.0	
cutting.edge.h_min_ratio	number		0.3	
cutting.materials	array (17)			the workpiece materials the setup offers, with their cutting data
cutting.materials[].key	string		"6061-T6"	
cutting.materials[].name	string		"Aluminum alloy 6061-T6"	
cutting.materials[].group	string		"VDI 22 / ISO N"	
cutting.materials[].density_g_cm3	number	g/cm³	2.7	
cutting.materials[].ktc_n_mm2	number	N/mm²	750.0	
cutting.materials[].krc_n_mm2	number	N/mm²	250.0	
cutting.materials[].kac_n_mm2	number	N/mm²	150.0	
cutting.materials[].kte_n_mm	number	N/mm	5.0	
cutting.materials[].kre_n_mm	number	N/mm	5.0	
cutting.materials[].kae_n_mm	number	N/mm	2.0	
cutting.materials[].kc_plunge_j_mm3	number	J/mm³	2.4	
cutting.materials[].vc_m_min	array (2)	m/min	[152, 335]	
cutting.materials[].alpha_um_m_k	number	µm/(m·K)	23.6	
cutting.materials[].c_j_kg_k	number	J/(kg·K)	896	
cutting.materials[].color	string		"#b9bdc3"	
cutting.materials[].source	string		"aluminium sharp-tool set (note C 1.5)"	
cutting.materials[].note	string		"measured sharp-carbide set (note C 1.5); Kienzle 800/0.2…	

key	type	unit	value	what it is
`cutting.materials[].oblique`	object			orthogonal cutting data: shear stress, shear and friction angles as functions of the chip, the speed and the rake; the reference chip and speed; the source
`cutting.materials[].oblique.tau_s_mpa`	array (4)	MPa	[205.928, 204.038, 0.016, -0.056]	
`cutting.materials[].oblique.phi_n_deg`	array (4)	°	[13.866, 62.929, 0.005, 0.259]	
`cutting.materials[].oblique.beta_a_deg`	array (4)	°	[20.835, -4.901, -0.007, 0.291]	
`cutting.materials[].oblique.h_ref_mm`	number	mm	0.02	
`cutting.materials[].oblique.vc_ref_m_min`	number	m/min	452.4	
`cutting.materials[].oblique.source`	string		"M. Ozsoy, MATECA 6(3) 263-271 (2025), Table 1 (orthogona…	
`cutting.materials[].life`	object			a carbide end mill's life in it for the wear book (wear.h): Taylor's pair and exponent, the flank wear criterion, judged by length or by VB; estimates where the source says est
`cutting.materials[].life.taylor_n`	number	N	0.4	
`cutting.materials[].life.vc_ref_m_min`	number	m/min	300.0	
`cutting.materials[].life.t_ref_min`	number		60.0	
`cutting.materials[].life.vb_c_mm`	number	mm	0.3	
`cutting.materials[].life.basis`	string		"length"	
`cutting.materials[].life.source`	string		"est: carbide start speeds 244-366 m/min, life set by BUE…	
`cutting.compliance`	object			the static give at the tool tip: the tool (modulus, core, flute length), the holder's joint, the frame, the drives by the nut's position
`cutting.compliance.note`	string		"static compliance at the tool tip, force on the tool: to…	
`cutting.compliance.tool_e_n_mm2`	number	N/mm²	600000.0	
`cutting.compliance.flute_core`	number		0.8	
`cutting.compliance.flute_length_d`	number		2.5	
`cutting.compliance.tool_density_kg_m3`	number		14500.0	the tool's mass for its own bending mode (cutting.dynamics): cemented carbide, the fluted section a share of the full circle's area (two to four flutes: about 0.6-0.75; assumed)
`cutting.compliance.flute_area`	number		0.65	
`cutting.compliance.joint_um_per_100n`	number	µm per 100 N	10.0	
`cutting.compliance.frame_um_per_100n`	array (3)	µm per 100 N		
`cutting.compliance.frame_portal_um_per_100n`	array (3)	µm per 100 N	[2.3495, 0.0, 0.0]	
`cutting.compliance.drive_um_per_100n`	object	µm per 100 N		each axis' drive, axially, by the nut's distance from the fixed bearing
`cutting.compliance.drive_um_per_100n.X`	array (2)			
`cutting.compliance.drive_um_per_100n.Y`	array (2)			
`cutting.compliance.drive_um_per_100n.Z`	array (2)			
`cutting.compliance.source`	string		"research note C 3.1-3.3; frame: out/fea/fea_result.json …	
`cutting.dynamics`	object			the tool point's frequency response for chatter: the structure's and the tool's natural frequencies and damping, the spindle's speed range, input shaping

key	type	unit	value	what it is
cutting.dynamics.note	string		"tool-point FRF, per direction x (feed axes X, Y): struct…	
cutting.dynamics.structure_fn_hz	array (2)	Hz	[161.0, 196.0]	
cutting.dynamics.tool_fn_hz	number	Hz	1500.0	
cutting.dynamics.zeta	array (3)		[0.03, 0.03, 0.03]	
cutting.dynamics.rpm_range	array (2)	rpm	[6000.0, 24000.0]	
cutting.dynamics.source	string		"research note C 4.1, 4.4 (ZOA; the holder mode's frequen…	
cutting.dynamics.input_shaping	object			the kernel's input shaping (K1.10, kernel/docs/08-input-shaping.md): what the structure modes ask of it
cutting.dynamics.input_shaping.note	string		"not needed at the commissioned accelerations ($120-$122 …	
cutting.dynamics.input_shaping.commissioned	object			the input shaping settings as commissioned
cutting.dynamics.input_shaping.commissioned.$1000	number		0	
cutting.dynamics.input_shaping.commissioned.$1001	number		0	
cutting.dynamics.input_shaping.commissioned.$1002	number		0	
cutting.dynamics.input_shaping.raised_acceleration	object			the input shaping settings for raised accelerations: the modes they suppress
cutting.dynamics.input_shaping.raised_acceleration.$1000	number		2	
cutting.dynamics.input_shaping.raised_acceleration.$1010	number		161.0	
cutting.dynamics.input_shaping.raised_acceleration.$1020	number		0.05	
cutting.dynamics.input_shaping.raised_acceleration.$1001	number		2	
cutting.dynamics.input_shaping.raised_acceleration.$1011	number		196.0	
cutting.dynamics.input_shaping.raised_acceleration.$1021	number		0.05	
cutting.dynamics.input_shaping.source	string		"kernel line K1.10 (0bd7ffe): tests/shaper_test.py residu…	

workholding

the fixtures: the tapped grid, vise, parallels, strap clamps, tape, V-blocks, friction

key	type	unit	value	what it is
workholding.grid	object			the table's tapped grid (model.py: M6 x 12 in columns GRID_XS either side, 9 rows from y -100)
workholding.grid.thread	string		"M6"	
workholding.grid.x	array (8)		[-125.0, -100.0, -75.0, -50.0, 50.0, 75.0, 100.0, 125.0]	
workholding.grid.y	array (9)		[-100.0, -75.0, -50.0, -25.0, 0.0, 25.0, 50.0, 75.0, 100.0]	
workholding.vise	object			a 75 mm self-centring vise (5th Axis V75100X class, note H 2.2 [R30]) on a 10 mm aluminium subplate bolted into the grid: the only low vise that leaves room under MC-1's 80 mm of Z (jaws 66 mm up)
workholding.vise.name	string		"75 mm self-centering precision vise (V75100X class) + 10…	
workholding.vise.self_centring	boolean		true	
workholding.vise.jaw_width	number		75.0	
workholding.vise.jaw_height	number		21.15	
workholding.vise.jaw_thickness	number		12.0	
workholding.vise.step_height	number		2.5	

key	type	unit	value	what it is
workholding.vise.min_opening	number		3.25	
workholding.vise.max_opening	number		77.0	
workholding.vise.base_height	number		35.0	
workholding.vise.body_width	number		75.0	
workholding.vise.body_length	number		100.0	
workholding.vise.plate_thickness	number		10.0	
workholding.vise.plate_length	number		130.0	
workholding.vise.plate_width	number		110.0	
workholding.vise.clamp_force_n	number	N	7400.0	
workholding.vise_low	object			the low-profile vise on the grid (a set-up's "vise": "low"): jaw tops low enough for a short probe to get over the blank with Z at its top
workholding.vise_low.name	string		"Low-profile vise, 75 mm jaws, 128 mm opening (Inception …	
workholding.vise_low.self_centring	boolean		false	
workholding.vise_low.jaw_width	number		75.0	
workholding.vise_low.jaw_height	number		15.0	
workholding.vise_low.jaw_thickness	number		15.0	
workholding.vise_low.step_height	number		3.0	
workholding.vise_low.min_opening	number		0.0	
workholding.vise_low.max_opening	number		128.0	
workholding.vise_low.base_height	number		5.0	
workholding.vise_low.body_width	number		75.0	
workholding.vise_low.body_length	number		210.0	
workholding.vise_low.plate_thickness	number		0.0	
workholding.vise_low.plate_length	number		0.0	
workholding.vise_low.plate_width	number		0.0	
workholding.vise_low.clamp_force_n	number	N	5000.0	
workholding.parallels	object			a standard parallel set (note H 2.6 [R45]): 1/8" thick, 1/2"..1-5/8" high in 1/8" steps, 6" long
workholding.parallels.thickness	number		3.18	
workholding.parallels.min_height	number		12.7	
workholding.parallels.step	number		3.175	
workholding.parallels.max_height	number		41.3	
workholding.parallels.length	number		152.4	
workholding.clamp	object			M6 serrated step clamp (Te-Co 30501M, 64 x 25 x 13) with a step block behind the stud; the stud screws into the grid; 5.4 N m gives 4.45 kN in the stud (Carr Lane, note H 2.3 [R37][R39])
workholding.clamp.strap_length	number		64.0	
workholding.clamp.strap_width	number		25.0	
workholding.clamp.strap_thickness	number		13.0	
workholding.clamp.nose_on_part	number		8.0	
workholding.clamp.heel_length	number		14.0	
workholding.clamp.reach_min	number		7.0	
workholding.clamp.reach_max	number		35.0	
workholding.clamp.stud_diameter	number		6.0	
workholding.clamp.washer_diameter	number		12.0	
workholding.clamp.nut_height	number		7.0	
workholding.clamp.bolt_preload_n	number	N	4450.0	

key	type	unit	value	what it is
workholding.tape	object			3M VHB 4910, 1.0 mm acrylic foam (note H 2.4 [R41]): 480 kPa overlap shear after 72 h; half of it counted (peel from the moment of a lateral force above the bond line, imperfect contact) [assumed]
workholding.tape.thickness	number		1.0	
workholding.tape.shear_mpa	number	MPa	0.48	
workholding.tape.effective_share	number		0.5	
workholding.vblock	object			a large V-block pair (Fowler X-Blox, 50.8 x 38.1 x 38.1, 38 mm capacity, 90 deg [R43]) with its strap clamp (force assumed)
workholding.vblock.length	number		38.1	
workholding.vblock.width	number		50.8	
workholding.vblock.height	number		38.1	
workholding.vblock.v_width	number		28.0	
workholding.vblock.clamp_force_n	number	N	2000.0	
workholding.friction	object			friction coefficients - the part on the table, in the jaws, under a strap - and the safety factor of the slip check
workholding.friction.part_on_table	number		0.3	physics, not design values (note H 2.7): the part slips when the cut beats these
workholding.friction.jaw_on_part	number		0.35	
workholding.friction.strap_on_part	number		0.35	
workholding.friction.safety_factor	number		3.0	

total_mass_kg

key	type	unit	value	what it is
total_mass_kg	number	kg	66.2	

controller

the controller: firmware, input filtering, boot time, the $-settings, the pin map

key	type	unit	value	what it is
controller.firmware	string		"Twinwright controller firmware (the kernel, 1.0) with it…	
controller.board_profile	string		"Teensy 4.1 + T41U5XBB: 24 MHz step timer, USB-CDC"	
controller.board	object			the controller board as the MCU side is modelled: step timer, step pulses and the $29 delay, input debounce, the sender's link (architecture 4.2)
controller.board.id	string		"teensy41_t41u5xbb"	
controller.board.text	string		"Teensy 4.1 (i.MX RT1062, Cortex-M7 600 MHz) on a Brookwo…	
controller.board.my_machine_board	string		"BOARD_T41U5XBB"	
controller.board.f_step_timer_hz	number	Hz	24000000	the step timer's clock (hal.f_step_timer); the twin's virtual clock runs at it
controller.board.timer_period_extra	number		1	
controller.board.timer_max_cycles	number		1048575	
controller.board.wake_up_ms	number	ms	2.0	
controller.board.step_pulse_latency_us	number	µs	0.2	
controller.board.step_toff_min_us	number	µs	2.0	
controller.board.pulse_timer_hz	number	Hz	150000000	
controller.board.signals	array (4)		["estop", "feed_hold", "cycle_start", "motor_fault"]	
controller.board.debounce_ms	number	ms	40	control inputs are acted on this long after their edge, if still on (0: at once)

key	type	unit	value	what it is
controller.board.debounced	array (4)		["reset", "feed_hold", "cycle_start", "safety_door"]	
controller.board.link_default	string		"usb_cdc"	the sender's line as the firmware is configured (my_machine.h USB_SERIAL_CDC): native USB, a UART, or Telnet
controller.board.links	object			the lines the board offers the sender: usb_cdc and telnet are packets the foreground polls, uart a byte per slot with a receive interrupt and the host's USB-serial adapter
controller.board.links.usb_cdc	object			the USB CDC link: frames, packets, buffers, polling, the host's latencies and jitter
controller.board.links.usb_cdc.frame_us	number	µs	125.0	
controller.board.links.usb_cdc.packet	number		512	
controller.board.links.usb_cdc.rx_capacity	number		4096	
controller.board.links.usb_cdc.poll_us	number	µs	50.0	
controller.board.links.usb_cdc.poll_bytes	number		20	
controller.board.links.usb_cdc.tx_buffer	number		6124	
controller.board.links.usb_cdc.tx_flush_us	number	µs	75.0	
controller.board.links.usb_cdc.host_out_latency_us	number	µs	250.0	
controller.board.links.usb_cdc.host_in_latency_us	number	µs	500.0	
controller.board.links.usb_cdc.jitter_us	number	µs	250.0	
controller.board.links.uart	object			the UART link through a USB-serial adapter: baud rate, the adapter's packets and timer, the host's latencies and jitter
controller.board.links.uart.baud	number		115200.0	
controller.board.links.uart.adapter	string		"FT232R-class USB-serial adapter"	
controller.board.links.uart.adapter_packet	number		62	
controller.board.links.uart.adapter_timer_ms	number	ms	16.0	the adapter's latency timer: a part-filled packet waits this long
controller.board.links.uart.host_out_latency_us	number	µs	1000.0	
controller.board.links.uart.host_in_latency_us	number	µs	1000.0	
controller.board.links.uart.jitter_us	number	µs	500.0	
controller.board.links.telnet	object			the network (Telnet) link: packets, buffers, polling, latencies and jitter
controller.board.links.telnet.frame_us	number	µs	0.0	
controller.board.links.telnet.packet	number		1460	
controller.board.links.telnet.rx_capacity	number		2920	
controller.board.links.telnet.poll_us	number	µs	1000.0	
controller.board.links.telnet.poll_bytes	number		1460	
controller.board.links.telnet.tx_buffer	number		1024	
controller.board.links.telnet.tx_flush_us	number	µs	1000.0	
controller.board.links.telnet.host_out_latency_us	number	µs	200.0	
controller.board.links.telnet.host_in_latency_us	number	µs	200.0	
controller.board.links.telnet.jitter_us	number	µs	300.0	
controller.board.fg_cost_us	object	µs		the foreground's CPU time per unit of work (a line parsed, a block planned, a step segment prepared), us

key	type	unit	value	what it is
controller.board.fg_cost_us.line	number		15.0	The foreground's CPU time (architecture 4.2 "前台耗时模型"): charged per unit of work - a line parsed and executed, a block planned (with the planner's recalculation), a step segment prepared - so a slow foreground starves the step segment buffer as on a real MCU. Estimates for the 600 MHz Cortex-M7 with its FPU (assumed): measure on the board with a GPIO toggled around each (03-commissioning).
controller.board.fg_cost_us.plan_block	number		5.0	
controller.board.fg_cost_us.segment	number		3.0	
controller.board.identity	object			What the controller says about itself in $I (research note I 1.2, docs/06 G3): the driver's own lines as the iMXRT1062 driver sets them (driver.c: hal.info, f_mcu, driver_version; USB_SERIAL_CDC 2: the options) and the board map's name - read the board's $I at commissioning and put its lines here (FREE MEMORY then too). twin_marker: one more line, [PLUGIN:<marker>], tells the twin apart; "" gives the board's $I exactly (a sender cannot tell them apart - docs/06 8.6: by choice only)
controller.board.identity.driver	string		"iMXRT1062"	
controller.board.identity.mcu_mhz	number	MHz	600	
controller.board.identity.driver_version	string		"260616"	
controller.board.identity.driver_options	string		"USB.2"	
controller.board.identity.board	string		"T41U5XBB"	
controller.board.identity.twin_marker	string		"MC-1 twin"	
controller.firmware_options	object			what the controller's firmware is built with: its build options, signals, probe, plugins, storage, axes, the driver's options
controller.firmware_options.my_machine_h	object			the build options as the firmware's my_machine.h names them
controller.firmware_options.my_machine_h.BOARD_T41U5XBB	number		1	
controller.firmware_options.my_machine_h.USB_SERIAL_CDC	number		2	
controller.firmware_options.my_machine_h.SPINDLE0_ENABLE	string		"SPINDLE_HUANYANG1"	
controller.firmware_options.my_machine_h.MODBUS_ENABLE	number		1	
controller.firmware_options.my_machine_h.ESTOP_ENABLE	number		1	
controller.firmware_options.my_machine_h.MOTOR_FAULT_ENABLE	number		1	
controller.firmware_options.my_machine_h.MPG_ENABLE	number		0	
controller.firmware_options.signals	array (4)		["estop", "feed_hold", "cycle_start", "motor_fault"]	
controller.firmware_options.probe	boolean		true	
controller.firmware_options.toolsetter	boolean		false	
controller.firmware_options.plugins	array (2)		["MODBUS", "HUANYANG VFD"]	
controller.firmware_options.nvs	string		"flash"	
controller.firmware_options.axes	number		3	
controller.firmware_options.driver	object			the board driver's options: debounce, reboot, clock, planner blocks, receive buffer
controller.firmware_options.driver.software_debounce	boolean		true	the iMXRT1062 driver's own (assumed from its source; the twin's driver the same): the control inputs are debounced in software (OPT S), $REBOOT restarts the MCU (NEWOPT REBOOT), the RT1062's RTC (NEWOPT RTC - though that driver never sets hal.driver_cap.rtc, so a real board answers $RTC with error:3: finding 94)
controller.firmware_options.driver.reboot	boolean		true	
controller.firmware_options.driver.rtc	boolean		true	

key	type	unit	value	what it is
controller.firmware_options.driver.planner_blocks	number		100	
controller.firmware_options.driver.rx_buffer	number		1024	
controller.panel	object			the control box's buttons and the controller inputs they are on
controller.panel.cycle_start	object			the cycle start button (SB2): how it reaches the controller, its input and pin, its contact
controller.panel.cycle_start.via	string		"pin"	
controller.panel.cycle_start.input	string		"CYCLE_START"	
controller.panel.cycle_start.teensy_pin	number		17	
controller.panel.cycle_start.button	string		"循环启动按钮 SB2（绿色，1 常开），电控箱门"	
controller.panel.cycle_start.contact	string		"NO"	
controller.panel.feed_hold	object			the feed hold button (SB3): how it reaches the controller, its input and pin, its contact
controller.panel.feed_hold.via	string		"pin"	
controller.panel.feed_hold.input	string		"FEED_HOLD"	
controller.panel.feed_hold.teensy_pin	number		16	
controller.panel.feed_hold.button	string		"进给保持按钮 SB3（黄色，1 常开），电控箱门"	
controller.panel.feed_hold.contact	string		"NO"	
controller.input_filter_us	number	µs	20.0	the board's isolated inputs: shorter pulses do not reach the pin (optocoupler; an RC filter would add to it). Limit inputs are not debounced in software (research F 2, 6)
controller.boot_ms	number	ms	50.0	
controller.settings	object			the $-settings (setting number -> value), also in mc1_settings.nc
controller.settings.$3	number		5	
controller.settings.$100	number		800	
controller.settings.$101	number		800	
controller.settings.$102	number		800	
controller.settings.$110	number		3000.0	
controller.settings.$111	number		3000.0	
controller.settings.$112	number		1500.0	
controller.settings.$120	number		300.0	
controller.settings.$121	number		300.0	
controller.settings.$122	number		200.0	
controller.settings.$130	number		202.0	
controller.settings.$131	number		152.0	
controller.settings.$132	number		82.0	
controller.settings.$10	number		15871	
controller.settings.$11	number		0.002	
controller.settings.$12	number		0.002	
controller.settings.$0	number		5.0	
controller.settings.$29	number		5.0	
controller.settings.$1	number		255	
controller.settings.$37	number		7	
controller.settings.$5	number		7	
controller.settings.$6	number		1	
controller.settings.$22	number		541	
controller.settings.$20	number		1	

key	type	unit	value	what it is
controller.settings.$21	number		1	
controller.settings.$23	number		0	
controller.settings.$44	number		4	
controller.settings.$45	number		3	
controller.settings.$46	number		0	
controller.settings.$180	number		50	
controller.settings.$181	number		50	
controller.settings.$182	number		50	
controller.settings.$190	number		1200	
controller.settings.$191	number		1200	
controller.settings.$192	number		600	
controller.settings.$26	number		250	
controller.settings.$27	number		2.0	
controller.settings.$43	number		1	
controller.settings.$65	number		32	
controller.settings.$341	number		3	
controller.settings.$342	number		70	
controller.settings.$343	number		25	
controller.settings.$344	number		200	
controller.settings.$346	number		0	
controller.settings.$30	number		24000	
controller.settings.$31	number		6000	
controller.settings.$340	number		5	
controller.settings.$374	number		3	
controller.settings.$375	number		50	
controller.settings.$394	number		15	
controller.settings.$460	number		1	
controller.settings_file	string		"twin/machine/mc1_settings.nc"	
controller.pin_map	array (16)			
controller.design_settings_table	array (28)			

`viewer`

the 3D views: the GLB model and its node groups, pose conventions, cable chains

key	type	unit	value	what it is
viewer.glb	string		"out/web/mc1.glb"	Everything twin/web needs to draw the machine: the GLB (moving groups) plus the drag chains, cable routes and Z service loop, which the viewer rebuilds per pose.
viewer.groups	object			the 3D model's motion groups: which part moves with which axis
viewer.groups.frame	string		"fixed_frame"	
viewer.groups.Y	string		"Y_table"	
viewer.groups.X	string		"X_carriage"	
viewer.groups.Z	string		"Z_slide"	
viewer.groups.dyn	string		"cable_chains"	
viewer.glb_pose	string		"model mm, Z up; exported at x_c = y_t = z_up = 0 where x…	
viewer.table_top_z	number		59.0	

key	type	unit	value	what it is
viewer.tool_tip	string		"spindle axis at model y = 0; tool tip = (x_c, 0, table_t…	
viewer.chains	object			the cable chains' geometry for the 3D view
viewer.chains.x	object			the X cable chain: link size, pitch, bend radius, length, where it is fixed, how it moves
viewer.chains.x.h	number		22.0	
viewer.chains.x.w	number		42.0	
viewer.chains.x.pitch	number		20.0	
viewer.chains.x.R	number		38.0	
viewer.chains.x.L	number		400.0	
viewer.chains.x.fixed	number		150.0	
viewer.chains.x.y	number		204.5	
viewer.chains.x.z	number		340.0	
viewer.chains.x.move_dx	number		40.0	
viewer.chains.x.lift	number		1.0	
viewer.chains.x.fill	number		0.85	
viewer.chains.x.loop_sign	number		-1	
viewer.chains.y	object			the Y cable chain: link size, pitch, bend radius, length, where it is fixed, how it moves
viewer.chains.y.h	number		15.0	
viewer.chains.y.w	number		22.0	
viewer.chains.y.pitch	number		15.0	
viewer.chains.y.R	number		20.0	
viewer.chains.y.L	number		360.0	
viewer.chains.y.fixed	number		150.0	
viewer.chains.y.x	number		-169.0	
viewer.chains.y.z	number		0.0	
viewer.chains.y.move_dy	number		0.0	
viewer.chains.y.lift	number		1.0	
viewer.chains.y.fill	number		0.85	
viewer.chains.y.loop_sign	number		-1	
viewer.routes	object			the cables' routes for the 3D view, as points (model mm)
viewer.routes.x_motor	array (5)			
viewer.routes.y_motor	array (6)			
viewer.routes.y_sensor_1	array (4)			
viewer.routes.y_sensor_2	array (4)			
viewer.routes.x_sensor_L	array (7)			
viewer.routes.x_sensor_R	array (7)			
viewer.routes.x_chain_in_zmotor	array (4)			
viewer.routes.x_chain_out	array (7)			
viewer.routes.y_chain_in_setter	array (5)			
viewer.routes.y_chain_out	array (6)			
viewer.route_groups	object			the routes that ride on a cable chain, and which

key	type	unit	value	what it is
viewer.route_groups.x_chain_in_zmotor	string		"x"	
viewer.route_groups.y_chain_in_setter	string		"y"	
viewer.cable_dia	object			each route's cable diameter (mm)
viewer.cable_dia.x_motor	number		9.0	
viewer.cable_dia.y_motor	number		9.0	
viewer.cable_dia.y_sensor_1	number		4.0	
viewer.cable_dia.y_sensor_2	number		4.0	
viewer.cable_dia.x_sensor_L	number		4.0	
viewer.cable_dia.x_sensor_R	number		4.0	
viewer.cable_dia.x_chain_in_zmotor	number		9.0	
viewer.cable_dia.x_chain_out	number		18.0	
viewer.cable_dia.y_chain_in_setter	number		4.0	
viewer.cable_dia.y_chain_out	number		4.0	
viewer.service_loop	object			the Z cables' service loop at three heights of Z, as points (model mm)
viewer.service_loop.0.0	array (7)			
viewer.service_loop.40.0	array (7)			
viewer.service_loop.80.0	array (7)			

provenance

where the values come from: a path pattern ("a[]" every item of a, "x/y_*" these keys, * a prefix; it covers what is under it; the most specific pattern wins) -> its record (B1)

key	t
provenance.axes[].travel	
provenance.axes[].travel.source	s
provenance.axes[].travel.note	s
provenance.axes[].travel.uncertainty	s

key	t
provenance.axes[].travel.refs	a (
provenance.axes[].travel.values	s
provenance.axes[].travel.history	a (
provenance.axes[].travel.history[].date	s
provenance.axes[].travel.history[].change	s
provenance.axes[].steps_per_mm	c
provenance.axes[].steps_per_mm.source	s
provenance.axes[].steps_per_mm.note	s
provenance.axes[].steps_per_mm.uncertainty	s
provenance.axes[].steps_per_mm.refs	a (
provenance.axes[].steps_per_mm.values	s
provenance.axes[].steps_per_mm.history	a (
provenance.axes[].steps_per_mm.history[].date	s
provenance.axes[].steps_per_mm.history[].change	s
provenance.axes[].switch.min/max	c
provenance.axes[].switch.min/max.source	s
provenance.axes[].switch.min/max.note	s
provenance.axes[].switch.min/max.uncertainty	s
provenance.axes[].switch.min/max.refs	a (
provenance.axes[].switch.min/max.values	s
provenance.axes[].switch.min/max.history	a (
provenance.axes[].switch.min/max.history[].date	s
provenance.axes[].switch.min/max.history[].change	s
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm	c
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm.source	s
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm.note	s
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm.uncertainty	s
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm.refs	a (
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm.values	s
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm.history	a (
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm.history[].date	s
provenance.axes[].switch.sn_mm/gap_mm/gap_design_mm/spot_r_cad_mm.history[].change	s

key	t
provenance.axes[].switch.lambda_mm/w_mm	c
provenance.axes[].switch.lambda_mm/w_mm.source	s
provenance.axes[].switch.lambda_mm/w_mm.note	s
provenance.axes[].switch.lambda_mm/w_mm.uncertainty	s
provenance.axes[].switch.lambda_mm/w_mm.refs	a (
provenance.axes[].switch.lambda_mm/w_mm.values	s
provenance.axes[].switch.lambda_mm/w_mm.history	a (
provenance.axes[].switch.lambda_mm/w_mm.history[].date	s
provenance.axes[].switch.lambda_mm/w_mm.history[].change	s
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm	c
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm.source	s
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm.note	s
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm.uncertainty	s
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm.refs	a (
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm.values	s
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm.history	a (
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm.history[].date	s
provenance.axes[].switch.hyst_mm/response_ms/jitter_ms/repeat_sigma_mm.history[].change	s
provenance.axes[].switch.temp_c/temp_coeff_per_k	c
provenance.axes[].switch.temp_c/temp_coeff_per_k.source	s
provenance.axes[].switch.temp_c/temp_coeff_per_k.note	s
provenance.axes[].switch.temp_c/temp_coeff_per_k.uncertainty	s
provenance.axes[].switch.temp_c/temp_coeff_per_k.refs	a (
provenance.axes[].switch.temp_c/temp_coeff_per_k.values	s
provenance.axes[].switch.temp_c/temp_coeff_per_k.history	a (
provenance.axes[].switch.temp_c/temp_coeff_per_k.history[].date	s
provenance.axes[].switch.temp_c/temp_coeff_per_k.history[].change	s
provenance.axes[].switch.ready_ms	c
provenance.axes[].switch.ready_ms.source	s
provenance.axes[].switch.ready_ms.note	s
provenance.axes[].switch.ready_ms.uncertainty	s
provenance.axes[].switch.ready_ms.refs	a (
provenance.axes[].switch.ready_ms.values	s
provenance.axes[].switch.ready_ms.history	a (
provenance.axes[].switch.ready_ms.history[].date	s
provenance.axes[].switch.ready_ms.history[].change	s
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr	c
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr.source	s
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr.note	s
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr.uncertainty	s

key	t
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr.refs	a (
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr.values	s
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr.history	a (
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr.history[].date	s
provenance.axes[].switch.contact/inputs/type/anchor/target/unit_sr.history[].change	s
provenance.axes[].hard_stop	o
provenance.axes[].hard_stop.source	s
provenance.axes[].hard_stop.note	s
provenance.axes[].hard_stop.uncertainty	s
provenance.axes[].hard_stop.refs	a (
provenance.axes[].hard_stop.values	s
provenance.axes[].hard_stop.history	a (
provenance.axes[].hard_stop.history[].date	s
provenance.axes[].hard_stop.history[].change	s
provenance.axes[].moving_mass_kg	o
provenance.axes[].moving_mass_kg.source	s
provenance.axes[].moving_mass_kg.note	s
provenance.axes[].moving_mass_kg.uncertainty	s
provenance.axes[].moving_mass_kg.refs	a (
provenance.axes[].moving_mass_kg.values	s
provenance.axes[].moving_mass_kg.history	a (
provenance.axes[].moving_mass_kg.history[].date	s
provenance.axes[].moving_mass_kg.history[].change	s
provenance.axes[].id/moves/home_dir	o
provenance.axes[].id/moves/home_dir.source	s
provenance.axes[].id/moves/home_dir.note	s
provenance.axes[].id/moves/home_dir.uncertainty	s
provenance.axes[].id/moves/home_dir.refs	a (
provenance.axes[].id/moves/home_dir.values	s
provenance.axes[].id/moves/home_dir.history	a (
provenance.axes[].id/moves/home_dir.history[].date	s
provenance.axes[].id/moves/home_dir.history[].change	s
provenance.axes[].dir_sign/direction	o
provenance.axes[].dir_sign/direction.source	s
provenance.axes[].dir_sign/direction.note	s
provenance.axes[].dir_sign/direction.uncertainty	s
provenance.axes[].dir_sign/direction.refs	a (
provenance.axes[].dir_sign/direction.values	s
provenance.axes[].dir_sign/direction.history	a (
provenance.axes[].dir_sign/direction.history[].date	s

key	t
provenance.axes[].dir_sign/direction.history[].change	s
provenance.sensor_materials	o
provenance.sensor_materials.source	s
provenance.sensor_materials.note	s
provenance.sensor_materials.uncertainty	s
provenance.sensor_materials.refs	a (
provenance.sensor_materials.values	s
provenance.sensor_materials.history	a (
provenance.sensor_materials.history[].date	s
provenance.sensor_materials.history[].change	s
provenance.wires	o
provenance.wires.source	s
provenance.wires.note	s
provenance.wires.uncertainty	s
provenance.wires.refs	a (
provenance.wires.values	s
provenance.wires.history	a (
provenance.wires.history[].date	s
provenance.wires.history[].change	s
provenance.faults	o
provenance.faults.source	s
provenance.faults.note	s
provenance.faults.uncertainty	s
provenance.faults.refs	a (
provenance.faults.values	s
provenance.faults.history	a (
provenance.faults.history[].date	s
provenance.faults.history[].change	s
provenance.controller.input_filter_us	o
provenance.controller.input_filter_us.source	s
provenance.controller.input_filter_us.note	s

key	t
provenance.controller.input_filter_us.uncertainty	s
provenance.controller.input_filter_us.refs	a (
provenance.controller.input_filter_us.values	s
provenance.controller.input_filter_us.history	a (
provenance.controller.input_filter_us.history[].date	s
provenance.controller.input_filter_us.history[].change	s
provenance.controller.boot_ms	c
provenance.controller.boot_ms.source	s
provenance.controller.boot_ms.note	s
provenance.controller.boot_ms.uncertainty	s
provenance.controller.boot_ms.refs	a (
provenance.controller.boot_ms.values	s
provenance.controller.boot_ms.history	a (
provenance.controller.boot_ms.history[].date	s
provenance.controller.boot_ms.history[].change	s
provenance.controller.board	c
provenance.controller.board.source	s
provenance.controller.board.note	s
provenance.controller.board.uncertainty	s
provenance.controller.board.refs	a (
provenance.controller.board.values	s
provenance.controller.board.history	a (
provenance.controller.board.history[].date	s
provenance.controller.board.history[].change	s
provenance.controller.board.links	c
provenance.controller.board.links.source	s
provenance.controller.board.links.note	s
provenance.controller.board.links.uncertainty	s
provenance.controller.board.links.refs	a (
provenance.controller.board.links.values	s
provenance.controller.board.links.history	a (
provenance.controller.board.links.history[].date	s
provenance.controller.board.links.history[].change	s
provenance.controller.board.identity	c
provenance.controller.board.identity.source	s
provenance.controller.board.identity.note	s
provenance.controller.board.identity.uncertainty	s
provenance.controller.board.identity.refs	a (
provenance.controller.board.identity.values	s
provenance.controller.board.identity.history	a (

key	t
provenance.controller.board.identity.history[].date	s
provenance.controller.board.identity.history[].change	s
provenance.controller.board.fg_cost_us	o
provenance.controller.board.fg_cost_us.source	s
provenance.controller.board.fg_cost_us.note	s
provenance.controller.board.fg_cost_us.uncertainty	s
provenance.controller.board.fg_cost_us.refs	a (
provenance.controller.board.fg_cost_us.values	s
provenance.controller.board.fg_cost_us.history	a (
provenance.controller.board.fg_cost_us.history[].date	s
provenance.controller.board.fg_cost_us.history[].change	s
provenance.controller.settings	o
provenance.controller.settings.source	s
provenance.controller.settings.note	s
provenance.controller.settings.uncertainty	s
provenance.controller.settings.refs	a (
provenance.controller.settings.values	s
provenance.controller.settings.history	a (
provenance.controller.settings.history[].date	s
provenance.controller.settings.history[].change	s
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file	o
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file.source	s
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file.note	s
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file.uncertainty	s
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file.refs	a (
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file.values	s
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file.history	a (
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file.history[].date	s
provenance.controller.firmware/board_profile/pin_map/design_settings_table/settings_file.history[].change	s
provenance.controller.firmware_options	o
provenance.controller.firmware_options.source	s
provenance.controller.firmware_options.note	s
provenance.controller.firmware_options.uncertainty	s
provenance.controller.firmware_options.refs	a (
provenance.controller.firmware_options.values	s
provenance.controller.firmware_options.history	a (
provenance.controller.firmware_options.history[].date	s
provenance.controller.firmware_options.history[].change	s
provenance.controller.firmware_options.driver	o
provenance.controller.firmware_options.driver.source	s
provenance.controller.firmware_options.driver.note	s

key	t
provenance.controller.firmware_options.driver.uncertainty	s
provenance.controller.firmware_options.driver.refs	a (
provenance.controller.firmware_options.driver.values	s
provenance.controller.firmware_options.driver.history	a (
provenance.controller.firmware_options.driver.history[].date	s
provenance.controller.firmware_options.driver.history[].change	s
provenance.controller.board.signals	o
provenance.controller.board.signals.source	s
provenance.controller.board.signals.note	s
provenance.controller.board.signals.uncertainty	s
provenance.controller.board.signals.refs	a (
provenance.controller.board.signals.values	s
provenance.controller.board.signals.history	a (
provenance.controller.board.signals.history[].date	s
provenance.controller.board.signals.history[].change	s
provenance.controller.panel	o
provenance.controller.panel.source	s
provenance.controller.panel.note	s
provenance.controller.panel.uncertainty	s
provenance.controller.panel.refs	a (
provenance.controller.panel.values	s
provenance.controller.panel.history	a (
provenance.controller.panel.history[].date	s
provenance.controller.panel.history[].change	s
provenance.supply	o
provenance.supply.source	s
provenance.supply.note	s
provenance.supply.uncertainty	s
provenance.supply.refs	a (
provenance.supply.values	s
provenance.supply.history	a (
provenance.supply.history[].date	s
provenance.supply.history[].change	s
provenance.power_on_pos	o
provenance.power_on_pos.source	s
provenance.power_on_pos.note	s
provenance.power_on_pos.uncertainty	s
provenance.power_on_pos.refs	a (
provenance.power_on_pos.values	s

key	t
provenance.power_on_pos.history	a (
provenance.power_on_pos.history[].date	s
provenance.power_on_pos.history[].change	s
provenance.drives.motor	o
provenance.drives.motor.source	s
provenance.drives.motor.note	s
provenance.drives.motor.uncertainty	s
provenance.drives.motor.refs	a (
provenance.drives.motor.values	s
provenance.drives.motor.history	a (
provenance.drives.motor.history[].date	s
provenance.drives.motor.history[].change	s
provenance.drives.motor.detent_nm	o
provenance.drives.motor.detent_nm.source	s
provenance.drives.motor.detent_nm.note	s
provenance.drives.motor.detent_nm.uncertainty	s
provenance.drives.motor.detent_nm.refs	a (
provenance.drives.motor.detent_nm.values	s
provenance.drives.motor.detent_nm.history	a (
provenance.drives.motor.detent_nm.history[].date	s
provenance.drives.motor.detent_nm.history[].change	s
provenance.drives.motor.torque_constant_nm_a	o
provenance.drives.motor.torque_constant_nm_a.source	s
provenance.drives.motor.torque_constant_nm_a.note	s
provenance.drives.motor.torque_constant_nm_a.uncertainty	s
provenance.drives.motor.torque_constant_nm_a.refs	a (
provenance.drives.motor.torque_constant_nm_a.values	s
provenance.drives.motor.torque_constant_nm_a.history	a (
provenance.drives.motor.torque_constant_nm_a.history[].date	s
provenance.drives.motor.torque_constant_nm_a.history[].change	s
provenance.drives.motor.iron_loss_nm_per_krpm	o
provenance.drives.motor.iron_loss_nm_per_krpm.source	s
provenance.drives.motor.iron_loss_nm_per_krpm.note	s
provenance.drives.motor.iron_loss_nm_per_krpm.uncertainty	s
provenance.drives.motor.iron_loss_nm_per_krpm.refs	a (
provenance.drives.motor.iron_loss_nm_per_krpm.values	s
provenance.drives.motor.iron_loss_nm_per_krpm.history	a (
provenance.drives.motor.iron_loss_nm_per_krpm.history[].date	s
provenance.drives.motor.iron_loss_nm_per_krpm.history[].change	s
provenance.drives.motor.power_up_jump	o

key	t
provenance.drives.motor.power_up_jump.source	s
provenance.drives.motor.power_up_jump.note	s
provenance.drives.motor.power_up_jump.uncertainty	s
provenance.drives.motor.power_up_jump.refs	a (
provenance.drives.motor.power_up_jump.values	s
provenance.drives.motor.power_up_jump.history	a (
provenance.drives.motor.power_up_jump.history[].date	s
provenance.drives.motor.power_up_jump.history[].change	s
provenance.drives.pullout_48v_rpm_nm	c
provenance.drives.pullout_48v_rpm_nm.source	s
provenance.drives.pullout_48v_rpm_nm.note	s
provenance.drives.pullout_48v_rpm_nm.uncertainty	s
provenance.drives.pullout_48v_rpm_nm.refs	a (
provenance.drives.pullout_48v_rpm_nm.values	s
provenance.drives.pullout_48v_rpm_nm.history	a (
provenance.drives.pullout_48v_rpm_nm.history[].date	s
provenance.drives.pullout_48v_rpm_nm.history[].change	s
provenance.drives.lead_mm/coupling_inertia_kgm2	c
provenance.drives.lead_mm/coupling_inertia_kgm2.source	s
provenance.drives.lead_mm/coupling_inertia_kgm2.note	s
provenance.drives.lead_mm/coupling_inertia_kgm2.uncertainty	s
provenance.drives.lead_mm/coupling_inertia_kgm2.refs	a (
provenance.drives.lead_mm/coupling_inertia_kgm2.values	s
provenance.drives.lead_mm/coupling_inertia_kgm2.history	a (
provenance.drives.lead_mm/coupling_inertia_kgm2.history[].date	s
provenance.drives.lead_mm/coupling_inertia_kgm2.history[].change	s
provenance.drives.model/note/step_s	c
provenance.drives.model/note/step_s.source	s
provenance.drives.model/note/step_s.note	s
provenance.drives.model/note/step_s.uncertainty	s
provenance.drives.model/note/step_s.refs	a (
provenance.drives.model/note/step_s.values	s
provenance.drives.model/note/step_s.history	a (
provenance.drives.model/note/step_s.history[].date	s
provenance.drives.model/note/step_s.history[].change	s
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n	c
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n.source	s
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n.note	s
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n.uncertainty	s
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n.refs	a (

key	t
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n.values	s
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n.history	a (
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n.history[].date	s
provenance.drives.axes[].id/screw_length_mm/screw_inertia_kgm2/rotating_inertia_kgm2/moving_mass_kg/gravity_n.history[].change	s
provenance.drives.axes[].equivalent_mass_kg	c
provenance.drives.axes[].equivalent_mass_kg.source	s
provenance.drives.axes[].equivalent_mass_kg.note	s
provenance.drives.axes[].equivalent_mass_kg.uncertainty	s
provenance.drives.axes[].equivalent_mass_kg.refs	a (
provenance.drives.axes[].equivalent_mass_kg.values	s
provenance.drives.axes[].equivalent_mass_kg.history	a (
provenance.drives.axes[].equivalent_mass_kg.history[].date	s
provenance.drives.axes[].equivalent_mass_kg.history[].change	s
provenance.drives.axes[].friction_n	c
provenance.drives.axes[].friction_n.source	s
provenance.drives.axes[].friction_n.note	s
provenance.drives.axes[].friction_n.uncertainty	s
provenance.drives.axes[].friction_n.refs	a (
provenance.drives.axes[].friction_n.values	s
provenance.drives.axes[].friction_n.history	a (
provenance.drives.axes[].friction_n.history[].date	s
provenance.drives.axes[].friction_n.history[].change	s
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg	c
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg.source	s
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg.note	s
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg.uncertainty	s
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg.refs	a (
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg.values	s
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg.history	a (
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg.history[].date	s
provenance.drives.axes[].l2.rotating_mass_kg/table_mass_kg/motor_mass_kg/screw_mass_kg.history[].change	s
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um	c
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um.source	s
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um.note	s
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um.uncertainty	s
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um.refs	a (
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um.values	s
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um.history	a (
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um.history[].date	s
provenance.drives.axes[].l2.k_axial_n_um/torsion_gj_nmm2/torsion_extra_mm/k_rest_n_um.history[].change	s

key	t
provenance.drives.axes[].l2.coupling_n_um	c
provenance.drives.axes[].l2.coupling_n_um.source	s
provenance.drives.axes[].l2.coupling_n_um.note	s
provenance.drives.axes[].l2.coupling_n_um.uncertainty	s
provenance.drives.axes[].l2.coupling_n_um.refs	a (
provenance.drives.axes[].l2.coupling_n_um.values	s
provenance.drives.axes[].l2.coupling_n_um.history	a (
provenance.drives.axes[].l2.coupling_n_um.history[].date	s
provenance.drives.axes[].l2.coupling_n_um.history[].change	s
provenance.drives.axes[].l2.nut_hertz_n_um15	c
provenance.drives.axes[].l2.nut_hertz_n_um15.source	s
provenance.drives.axes[].l2.nut_hertz_n_um15.note	s
provenance.drives.axes[].l2.nut_hertz_n_um15.uncertainty	s
provenance.drives.axes[].l2.nut_hertz_n_um15.refs	a (
provenance.drives.axes[].l2.nut_hertz_n_um15.values	s
provenance.drives.axes[].l2.nut_hertz_n_um15.history	a (
provenance.drives.axes[].l2.nut_hertz_n_um15.history[].date	s
provenance.drives.axes[].l2.nut_hertz_n_um15.history[].change	s
provenance.drives.axes[].l2.frame	c
provenance.drives.axes[].l2.frame.source	s
provenance.drives.axes[].l2.frame.note	s
provenance.drives.axes[].l2.frame.uncertainty	s
provenance.drives.axes[].l2.frame.refs	a (
provenance.drives.axes[].l2.frame.values	s
provenance.drives.axes[].l2.frame.history	a (
provenance.drives.axes[].l2.frame.history[].date	s
provenance.drives.axes[].l2.frame.history[].change	s
provenance.drives.axes[].l2.friction_motor_n/friction_table_n	c
provenance.drives.axes[].l2.friction_motor_n/friction_table_n.source	s
provenance.drives.axes[].l2.friction_motor_n/friction_table_n.note	s
provenance.drives.axes[].l2.friction_motor_n/friction_table_n.uncertainty	s
provenance.drives.axes[].l2.friction_motor_n/friction_table_n.refs	a (
provenance.drives.axes[].l2.friction_motor_n/friction_table_n.values	s
provenance.drives.axes[].l2.friction_motor_n/friction_table_n.history	a (
provenance.drives.axes[].l2.friction_motor_n/friction_table_n.history[].date	s
provenance.drives.axes[].l2.friction_motor_n/friction_table_n.history[].change	s
provenance.drives.drive	c
provenance.drives.drive.source	s
provenance.drives.drive.note	s
provenance.drives.drive.uncertainty	s

key	t
provenance.drives.drive.refs	a (
provenance.drives.drive.values	s
provenance.drives.drive.history	a (
provenance.drives.drive.history[].date	s
provenance.drives.drive.history[].change	s
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg	c
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg.source	s
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg.note	s
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg.uncertainty	s
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg.refs	a (
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg.values	s
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg.history	a (
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg.history[].date	s
provenance.drives.drive.open_loop_below_rpm/current_loop_hz/bus_capacitance_uf/overvoltage_v/max_phase_advance_deg.history[].change	s
provenance.drives.drive.l3_sub_steps	c
provenance.drives.drive.l3_sub_steps.source	s
provenance.drives.drive.l3_sub_steps.note	s
provenance.drives.drive.l3_sub_steps.uncertainty	s
provenance.drives.drive.l3_sub_steps.refs	a (
provenance.drives.drive.l3_sub_steps.values	s
provenance.drives.drive.l3_sub_steps.history	a (
provenance.drives.drive.l3_sub_steps.history[].date	s
provenance.drives.drive.l3_sub_steps.history[].change	s
provenance.drives.alm	c
provenance.drives.alm.source	s
provenance.drives.alm.note	s
provenance.drives.alm.uncertainty	s
provenance.drives.alm.refs	a (
provenance.drives.alm.values	s
provenance.drives.alm.history	a (
provenance.drives.alm.history[].date	s
provenance.drives.alm.history[].change	s
provenance.drives.power	c
provenance.drives.power.source	s
provenance.drives.power.note	s
provenance.drives.power.uncertainty	s
provenance.drives.power.refs	a (
provenance.drives.power.values	s
provenance.drives.power.history	a (
provenance.drives.power.history[].date	s

key	t
provenance.drives.power.history[].change	s
provenance.drives.l2	o
provenance.drives.l2.source	s
provenance.drives.l2.note	s
provenance.drives.l2.uncertainty	s
provenance.drives.l2.refs	a (
provenance.drives.l2.values	s
provenance.drives.l2.history	a (
provenance.drives.l2.history[].date	s
provenance.drives.l2.history[].change	s
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s	o
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s.source	s
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s.note	s
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s.uncertainty	s
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s.refs	a (
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s.values	s
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s.history	a (
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s.history[].date	s
provenance.drives.l2.zeta_axial/zeta_coupling/servo_damping_ratio/current_lag_ms/smoothing_ms/contact_restitution/impact_speed_mm_s.history[].change	s
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um	o
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um.source	s
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um.note	s
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um.uncertainty	s
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um.refs	a (
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um.values	s
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um.history	a (
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um.history[].date	s
provenance.drives.l2.stribeck_ratio/stribeck_v_mm_s/viscous_n_s_mm/lugre_sigma0_n_um.history[].change	s
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k	o
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k.source	s
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k.note	s
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k.uncertainty	s
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k.refs	a (
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k.values	s
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k.history	a (
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k.history[].date	s
provenance.cutting.materials.alpha_um_m_k/c_j_kg_k.history[].change	s
provenance.spindle.vfd.params	o
provenance.spindle.vfd.params.source	s
provenance.spindle.vfd.params.note	s

key	t
provenance.spindle.vfd.params.uncertainty	s
provenance.spindle.vfd.params.refs	a (
provenance.spindle.vfd.params.values	s
provenance.spindle.vfd.params.history	a (
provenance.spindle.vfd.params.history[].date	s
provenance.spindle.vfd.params.history[].change	s
provenance.spindle.vfd.factory	o
provenance.spindle.vfd.factory.source	s
provenance.spindle.vfd.factory.note	s
provenance.spindle.vfd.factory.uncertainty	s
provenance.spindle.vfd.factory.refs	a (
provenance.spindle.vfd.factory.values	s
provenance.spindle.vfd.factory.history	a (
provenance.spindle.vfd.factory.history[].date	s
provenance.spindle.vfd.factory.history[].change	s
provenance.spindle.vfd.protocol	o
provenance.spindle.vfd.protocol.source	s
provenance.spindle.vfd.protocol.note	s
provenance.spindle.vfd.protocol.uncertainty	s
provenance.spindle.vfd.protocol.refs	a (
provenance.spindle.vfd.protocol.values	s
provenance.spindle.vfd.protocol.history	a (
provenance.spindle.vfd.protocol.history[].date	s
provenance.spindle.vfd.protocol.history[].change	s
provenance.spindle.vfd.reply_latency_ms/min_gap_ms	o
provenance.spindle.vfd.reply_latency_ms/min_gap_ms.source	s
provenance.spindle.vfd.reply_latency_ms/min_gap_ms.note	s
provenance.spindle.vfd.reply_latency_ms/min_gap_ms.uncertainty	s
provenance.spindle.vfd.reply_latency_ms/min_gap_ms.refs	a (
provenance.spindle.vfd.reply_latency_ms/min_gap_ms.values	s
provenance.spindle.vfd.reply_latency_ms/min_gap_ms.history	a (
provenance.spindle.vfd.reply_latency_ms/min_gap_ms.history[].date	s
provenance.spindle.vfd.reply_latency_ms/min_gap_ms.history[].change	s
provenance.spindle.vfd.boot_s/oc_factor/overload_*	o
provenance.spindle.vfd.boot_s/oc_factor/overload_*.source	s
provenance.spindle.vfd.boot_s/oc_factor/overload_*.note	s
provenance.spindle.vfd.boot_s/oc_factor/overload_*.uncertainty	s
provenance.spindle.vfd.boot_s/oc_factor/overload_*.refs	a (
provenance.spindle.vfd.boot_s/oc_factor/overload_*.values	s
provenance.spindle.vfd.boot_s/oc_factor/overload_*.history	a (

key	t
provenance.spindle.vfd.boot_s/oc_factor/overload_*.history[].date	s
provenance.spindle.vfd.boot_s/oc_factor/overload_*.history[].change	s
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v	c
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v.source	s
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v.note	s
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v.uncertainty	s
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v.refs	a (
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v.values	s
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v.history	a (
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v.history[].date	s
provenance.spindle.vfd.bus_capacitance_uf/lu_v/control_off_v/control_w/dc_bus_v.history[].change	s
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port	c
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port.source	s
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port.note	s
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port.uncertainty	s
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port.refs	a (
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port.values	s
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port.history	a (
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port.history[].date	s
provenance.spindle.vfd.model/rated_power_w/rated_current_a/supply/port.history[].change	s
provenance.spindle.motor.model/rated_*/poles	c
provenance.spindle.motor.model/rated_*/poles.source	s
provenance.spindle.motor.model/rated_*/poles.note	s
provenance.spindle.motor.model/rated_*/poles.uncertainty	s
provenance.spindle.motor.model/rated_*/poles.refs	a (
provenance.spindle.motor.model/rated_*/poles.values	s
provenance.spindle.motor.model/rated_*/poles.history	a (
provenance.spindle.motor.model/rated_*/poles.history[].date	s
provenance.spindle.motor.model/rated_*/poles.history[].change	s
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a	c
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a.source	s
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a.note	s
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a.uncertainty	s
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a.refs	a (
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a.values	s
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a.history	a (
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a.history[].date	s
provenance.spindle.motor.rated_slip/breakdown_torque_ratio/no_load_current_a.history[].change	s
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms	c
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms.source	s

key	t
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms.note	s
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms.uncertainty	s
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms.refs	a (
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms.values	s
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms.history	a (
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms.history[].date	s
provenance.spindle.motor.inertia_kgm2/friction_nm/viscous_nms.history[].change	s
provenance.spindle.coolant_pump	o
provenance.spindle.coolant_pump.source	s
provenance.spindle.coolant_pump.note	s
provenance.spindle.coolant_pump.uncertainty	s
provenance.spindle.coolant_pump.refs	a (
provenance.spindle.coolant_pump.values	s
provenance.spindle.coolant_pump.history	a (
provenance.spindle.coolant_pump.history[].date	s
provenance.spindle.coolant_pump.history[].change	s
provenance.spindle.encoder	o
provenance.spindle.encoder.source	s
provenance.spindle.encoder.note	s
provenance.spindle.encoder.uncertainty	s
provenance.spindle.encoder.refs	a (
provenance.spindle.encoder.values	s
provenance.spindle.encoder.history	a (
provenance.spindle.encoder.history[].date	s
provenance.spindle.encoder.history[].change	s
provenance.tool_setter	o
provenance.tool_setter.source	s
provenance.tool_setter.note	s
provenance.tool_setter.uncertainty	s
provenance.tool_setter.refs	a (
provenance.tool_setter.values	s
provenance.tool_setter.history	a (
provenance.tool_setter.history[].date	s
provenance.tool_setter.history[].change	s
provenance.tool_setter.overtravel/contact	o
provenance.tool_setter.overtravel/contact.source	s
provenance.tool_setter.overtravel/contact.note	s
provenance.tool_setter.overtravel/contact.uncertainty	s
provenance.tool_setter.overtravel/contact.refs	a (
provenance.tool_setter.overtravel/contact.values	s

key	t
provenance.tool_setter.overtravel/contact.history	a (
provenance.tool_setter.overtravel/contact.history[].date	s
provenance.tool_setter.overtravel/contact.history[].change	s
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms	c
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms.source	s
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms.note	s
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms.uncertainty	s
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms.refs	a (
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms.values	s
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms.history	a (
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms.history[].date	s
provenance.tool_setter.pretravel_mm/repeat_sigma_mm/reset_hyst_mm/bounces/bounce_ms.history[].change	s
provenance.touch_probe.model	c
provenance.touch_probe.model.source	s
provenance.touch_probe.model.note	s
provenance.touch_probe.model.uncertainty	s
provenance.touch_probe.model.refs	a (
provenance.touch_probe.model.values	s
provenance.touch_probe.model.history	a (
provenance.touch_probe.model.history[].date	s
provenance.touch_probe.model.history[].change	s
provenance.touch_probe.stylus_travel_mm	c
provenance.touch_probe.stylus_travel_mm.source	s
provenance.touch_probe.stylus_travel_mm.note	s
provenance.touch_probe.stylus_travel_mm.uncertainty	s
provenance.touch_probe.stylus_travel_mm.refs	a (
provenance.touch_probe.stylus_travel_mm.values	s
provenance.touch_probe.stylus_travel_mm.history	a (
provenance.touch_probe.stylus_travel_mm.history[].date	s
provenance.touch_probe.stylus_travel_mm.history[].change	s
provenance.touch_probe.geometry	c
provenance.touch_probe.geometry.source	s
provenance.touch_probe.geometry.note	s
provenance.touch_probe.geometry.uncertainty	s
provenance.touch_probe.geometry.refs	a (
provenance.touch_probe.geometry.values	s
provenance.touch_probe.geometry.history	a (
provenance.touch_probe.geometry.history[].date	s
provenance.touch_probe.geometry.history[].change	s
provenance.touch_probe.styli	c

key	t
provenance.touch_probe.styli.source	s
provenance.touch_probe.styli.note	s
provenance.touch_probe.styli.uncertainty	s
provenance.touch_probe.styli.refs	a (
provenance.touch_probe.styli.values	s
provenance.touch_probe.styli.history	a (
provenance.touch_probe.styli.history[].date	s
provenance.touch_probe.styli.history[].change	s
provenance.touch_probe.typical	o
provenance.touch_probe.typical.source	s
provenance.touch_probe.typical.note	s
provenance.touch_probe.typical.uncertainty	s
provenance.touch_probe.typical.refs	a (
provenance.touch_probe.typical.values	s
provenance.touch_probe.typical.history	a (
provenance.touch_probe.typical.history[].date	s
provenance.touch_probe.typical.history[].change	s
provenance.sound.sources	o
provenance.sound.sources.source	s
provenance.sound.sources.note	s
provenance.sound.sources.uncertainty	s
provenance.sound.sources.refs	a (
provenance.sound.sources.values	s
provenance.sound.sources.history	a (
provenance.sound.sources.history[].date	s
provenance.sound.sources.history[].change	s
provenance.sound.sources.pump	o
provenance.sound.sources.pump.source	s
provenance.sound.sources.pump.note	s
provenance.sound.sources.pump.uncertainty	s
provenance.sound.sources.pump.refs	a (
provenance.sound.sources.pump.values	s
provenance.sound.sources.pump.history	a (
provenance.sound.sources.pump.history[].date	s
provenance.sound.sources.pump.history[].change	s
provenance.sound.listener	o
provenance.sound.listener.source	s
provenance.sound.listener.note	s
provenance.sound.listener.uncertainty	s

key	t
provenance.sound.listener.refs	a (
provenance.sound.listener.values	s
provenance.sound.listener.history	a (
provenance.sound.listener.history[].date	s
provenance.sound.listener.history[].change	s
provenance.sound.radiation	o
provenance.sound.radiation.source	s
provenance.sound.radiation.note	s
provenance.sound.radiation.uncertainty	s
provenance.sound.radiation.refs	a (
provenance.sound.radiation.values	s
provenance.sound.radiation.history	a (
provenance.sound.radiation.history[].date	s
provenance.sound.radiation.history[].change	s
provenance.collision	o
provenance.collision.source	s
provenance.collision.note	s
provenance.collision.uncertainty	s
provenance.collision.refs	a (
provenance.collision.values	s
provenance.collision.history	a (
provenance.collision.history[].date	s
provenance.collision.history[].change	s
provenance.collision.groups	o
provenance.collision.groups.source	s
provenance.collision.groups.note	s
provenance.collision.groups.uncertainty	s
provenance.collision.groups.refs	a (
provenance.collision.groups.values	s
provenance.collision.groups.history	a (
provenance.collision.groups.history[].date	s
provenance.collision.groups.history[].change	s
provenance.collision.near_miss_mm	o
provenance.collision.near_miss_mm.source	s
provenance.collision.near_miss_mm.note	s
provenance.collision.near_miss_mm.uncertainty	s
provenance.collision.near_miss_mm.refs	a (
provenance.collision.near_miss_mm.values	s
provenance.collision.near_miss_mm.history	a (
provenance.collision.near_miss_mm.history[].date	s

key	t
provenance.collision.near_miss_mm.history[].change	s
provenance.cutting.materials	o
provenance.cutting.materials.source	s
provenance.cutting.materials.note	s
provenance.cutting.materials.uncertainty	s
provenance.cutting.materials.refs	a (
provenance.cutting.materials.values	s
provenance.cutting.materials.history	a (
provenance.cutting.materials.history[].date	s
provenance.cutting.materials.history[].change	s
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3	o
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3.source	s
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3.note	s
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3.uncertainty	s
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3.refs	a (
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3.values	s
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3.history	a (
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3.history[].date	s
provenance.cutting.material/ktc_n_mm2/krc_n_mm2/kac_n_mm2/kte_n_mm/kre_n_mm/kae_n_mm/kc_plunge_j_mm3.history[].change	s
provenance.cutting.tool_default/tool_geometry	o
provenance.cutting.tool_default/tool_geometry.source	s
provenance.cutting.tool_default/tool_geometry.note	s
provenance.cutting.tool_default/tool_geometry.uncertainty	s
provenance.cutting.tool_default/tool_geometry.refs	a (
provenance.cutting.tool_default/tool_geometry.values	s
provenance.cutting.tool_default/tool_geometry.history	a (
provenance.cutting.tool_default/tool_geometry.history[].date	s
provenance.cutting.tool_default/tool_geometry.history[].change	s
provenance.cutting.tool_strength	o
provenance.cutting.tool_strength.source	s
provenance.cutting.tool_strength.note	s
provenance.cutting.tool_strength.uncertainty	s
provenance.cutting.tool_strength.refs	a (
provenance.cutting.tool_strength.values	s
provenance.cutting.tool_strength.history	a (
provenance.cutting.tool_strength.history[].date	s
provenance.cutting.tool_strength.history[].change	s
provenance.cutting.force_model	o
provenance.cutting.force_model.source	s
provenance.cutting.force_model.note	s

key	t
provenance.cutting.force_model.uncertainty	s
provenance.cutting.force_model.refs	a (
provenance.cutting.force_model.values	s
provenance.cutting.force_model.history	a (
provenance.cutting.force_model.history[].date	s
provenance.cutting.force_model.history[].change	s
provenance.cutting.runout	o
provenance.cutting.runout.source	s
provenance.cutting.runout.note	s
provenance.cutting.runout.uncertainty	s
provenance.cutting.runout.refs	a (
provenance.cutting.runout.values	s
provenance.cutting.runout.history	a (
provenance.cutting.runout.history[].date	s
provenance.cutting.runout.history[].change	s
provenance.cutting.edge	o
provenance.cutting.edge.source	s
provenance.cutting.edge.note	s
provenance.cutting.edge.uncertainty	s
provenance.cutting.edge.refs	a (
provenance.cutting.edge.values	s
provenance.cutting.edge.history	a (
provenance.cutting.edge.history[].date	s
provenance.cutting.edge.history[].change	s
provenance.cutting.wear	o
provenance.cutting.wear.source	s
provenance.cutting.wear.note	s
provenance.cutting.wear.uncertainty	s
provenance.cutting.wear.refs	a (
provenance.cutting.wear.values	s
provenance.cutting.wear.history	a (
provenance.cutting.wear.history[].date	s
provenance.cutting.wear.history[].change	s
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach	o
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach.source	s
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach.note	s
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach.uncertainty	s
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach.refs	a (
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach.values	s

key	t
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach.history	a (
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach.history[].date	s
provenance.cutting.stock_res_mm/chunk_mm/chunk_ms/force_step/merge_tol_um/pyramid/sweep_reach.history[].change	s
provenance.cutting.variation	o
provenance.cutting.variation.source	s
provenance.cutting.variation.note	s
provenance.cutting.variation.uncertainty	s
provenance.cutting.variation.refs	a (
provenance.cutting.variation.values	s
provenance.cutting.variation.history	a (
provenance.cutting.variation.history[].date	s
provenance.cutting.variation.history[].change	s
provenance.cutting.compliance	o
provenance.cutting.compliance.source	s
provenance.cutting.compliance.note	s
provenance.cutting.compliance.uncertainty	s
provenance.cutting.compliance.refs	a (
provenance.cutting.compliance.values	s
provenance.cutting.compliance.history	a (
provenance.cutting.compliance.history[].date	s
provenance.cutting.compliance.history[].change	s
provenance.cutting.dynamics	o
provenance.cutting.dynamics.source	s
provenance.cutting.dynamics.note	s
provenance.cutting.dynamics.uncertainty	s
provenance.cutting.dynamics.refs	a (
provenance.cutting.dynamics.values	s
provenance.cutting.dynamics.history	a (
provenance.cutting.dynamics.history[].date	s
provenance.cutting.dynamics.history[].change	s
provenance.workholding	o
provenance.workholding.source	s
provenance.workholding.note	s
provenance.workholding.uncertainty	s
provenance.workholding.refs	a (
provenance.workholding.values	s
provenance.workholding.history	a (
provenance.workholding.history[].date	s
provenance.workholding.history[].change	s
provenance.workholding.vise_low	o

key	t
provenance.workholding.vise_low.source	s
provenance.workholding.vise_low.note	s
provenance.workholding.vise_low.uncertainty	s
provenance.workholding.vise_low.refs	a (
provenance.workholding.vise_low.values	s
provenance.workholding.vise_low.history	a (
provenance.workholding.vise_low.history[].date	s
provenance.workholding.vise_low.history[].change	s
provenance.total_mass_kg	c
provenance.total_mass_kg.source	s
provenance.total_mass_kg.note	s
provenance.total_mass_kg.uncertainty	s
provenance.total_mass_kg.refs	a (
provenance.total_mass_kg.values	s
provenance.total_mass_kg.history	a (
provenance.total_mass_kg.history[].date	s
provenance.total_mass_kg.history[].change	s
provenance.viewer	c
provenance.viewer.source	s
provenance.viewer.note	s
provenance.viewer.uncertainty	s
provenance.viewer.refs	a (
provenance.viewer.values	s
provenance.viewer.history	a (
provenance.viewer.history[].date	s
provenance.viewer.history[].change	s
provenance.schema/generated/units/name/coordinates	c
provenance.schema/generated/units/name/coordinates.source	s
provenance.schema/generated/units/name/coordinates.note	s
provenance.schema/generated/units/name/coordinates.uncertainty	s
provenance.schema/generated/units/name/coordinates.refs	a (
provenance.schema/generated/units/name/coordinates.values	s

key	t
provenance.schema/generated/units/name/coordinates.history	a (
provenance.schema/generated/units/name/coordinates.history[].date	s
provenance.schema/generated/units/name/coordinates.history[].change	s

provenance.accuracy

Where the twin's internal error-model parameters come from (the error model is off by default); the records' values are not published.

key	type	unit	what it is
provenance.accuracy	object		
provenance.accuracy.source	string		the category: design, cad (off the CAD model), datasheet, literature (research notes, catalogues), calc (from other values), assumption, measured (on this machine), model (a simulation setting)
provenance.accuracy.note	string		where exactly, and what to measure
provenance.accuracy.uncertainty	string		how far off the values may be, in words ("+-20 %", "exact")
provenance.accuracy.refs	array (3)		references: research notes, datasheets, source files
provenance.accuracy.values	string		a digest of the values the pattern covers: when they change, the history gets an item
provenance.accuracy.history	array (3)		the entry's changes, dated (source, note, uncertainty, values)
provenance.accuracy.history[].date	string		
provenance.accuracy.history[].change	string		
provenance.accuracy.thermal.frame	object		
provenance.accuracy.thermal.frame.source	string		
provenance.accuracy.thermal.frame.note	string		
provenance.accuracy.thermal.frame.uncertainty	string		
provenance.accuracy.thermal.frame.refs	array (2)		
provenance.accuracy.thermal.frame.values	string		
provenance.accuracy.thermal.frame.history	array (1)		
provenance.accuracy.thermal.frame.history[].date	string		
provenance.accuracy.thermal.frame.history[].change	string		
provenance.accuracy.thermal.ambient_dt_k	object		
provenance.accuracy.thermal.ambient_dt_k.source	string		
provenance.accuracy.thermal.ambient_dt_k.note	string		
provenance.accuracy.thermal.ambient_dt_k.uncertainty	string		
provenance.accuracy.thermal.ambient_dt_k.refs	array (0)		
provenance.accuracy.thermal.ambient_dt_k.values	string		
provenance.accuracy.thermal.ambient_dt_k.history	array (1)		
provenance.accuracy.thermal.ambient_dt_k.history[].date	string		
provenance.accuracy.thermal.ambient_dt_k.history[].change	string		
provenance.accuracy.thermal.part	object		
provenance.accuracy.thermal.part.source	string		
provenance.accuracy.thermal.part.note	string		
provenance.accuracy.thermal.part.uncertainty	string		
provenance.accuracy.thermal.part.refs	array (1)		
provenance.accuracy.thermal.part.values	string		
provenance.accuracy.thermal.part.history	array (1)		

key	type	unit	what it is
`provenance.accuracy.thermal.part.history[].date`	string		
`provenance.accuracy.thermal.part.history[].change`	string		
`provenance.accuracy.thermal.dynamic`	object		
`provenance.accuracy.thermal.dynamic.source`	string		
`provenance.accuracy.thermal.dynamic.note`	string		
`provenance.accuracy.thermal.dynamic.uncertainty`	string		
`provenance.accuracy.thermal.dynamic.refs`	array (1)		
`provenance.accuracy.thermal.dynamic.values`	string		
`provenance.accuracy.thermal.dynamic.history`	array (1)		
`provenance.accuracy.thermal.dynamic.history[].date`	string		
`provenance.accuracy.thermal.dynamic.history[].change`	string		

`provenance.drives.axes[].l2.coupling_play_um`

Where the coupling's play comes from; the record's values are not published.

key	type	unit	what it is
`provenance.drives.axes[].l2.coupling_play_um`	object		
`provenance.drives.axes[].l2.coupling_play_um.source`	string		the category: design, cad (off the CAD model), datasheet, literature (research notes, catalogues), calc (from other values), assumption, measured (on this machine), model (a simulation setting)
`provenance.drives.axes[].l2.coupling_play_um.note`	string		where exactly, and what to measure
`provenance.drives.axes[].l2.coupling_play_um.uncertainty`	string		how far off the values may be, in words ("+-20 %", "exact")
`provenance.drives.axes[].l2.coupling_play_um.refs`	array (1)		references: research notes, datasheets, source files
`provenance.drives.axes[].l2.coupling_play_um.values`	string		a digest of the values the pattern covers: when they change, the history gets an item
`provenance.drives.axes[].l2.coupling_play_um.history`	array (2)		the entry's changes, dated (source, note, uncertainty, values)
`provenance.drives.axes[].l2.coupling_play_um.history[].date`	string		
`provenance.drives.axes[].l2.coupling_play_um.history[].change`	string		

1820 keys.

Protokoll

Twinwrights eigener Kernel (die Steuerung) spricht das serielle Protokoll von Grbl 1.1, mit den Erweiterungen, die aktuelle Grbl-Sender-Programme verwenden. Ein Sender-Programm, das für Grbl geschrieben wurde, funktioniert damit. Die geprüften Sender-Programme und ihre Einstellungen stehen in G-Code-Sender-Programme.

Verbinden

Wo	Wie
Der Zwilling in Studio	TCP-Port 2323 auf diesem Computer, jeweils ein Sender-Programm.
Die MC-1	Ihr nativer USB-Port, ein serieller Port (COM*n* unter Windows). Die Baudrate spielt keine Rolle; verwenden Sie 115200, wo ein Sender-Programm eine verlangt. Die Steuerung antwortet, sobald das Sender-Programm DTR setzt.

Zeilen und Antworten

- **Zeilenende:** Eine Zeile endet mit LF oder CR. Ein Paar aus CR LF oder LF CR zählt einmal; zwei LF sind zwei Zeilen.
- **Antworten:** Die Steuerung antwortet auf jede Zeile mit `ok` oder `error:N`, in der Reihenfolge, in der die Zeilen kamen. Fehlercodes listet jedes N auf.
- **Zeilenlänge:** bis zu 256 Zeichen. Eine längere Zeile wird mit `error:11` beantwortet.
- **Zeichenzählung:** Der Empfangspuffer fasst 1024 Bytes. Ein Sender-Programm darf so viele unbeantwortete Bytes unterwegs haben. `Bf:` im Statusbericht gibt den freien Platz an.
- **Planer:** 100 Blöcke. `Bf:` gibt auch die freien Blöcke an.
- **Wartende Zeilen:** Bei Vorschub halt oder geöffneter Schutztür warten Zeilen im Puffer. Sie werden beantwortet, sobald der Zyklus wieder startet. Ein Reset, `0x19` (Stopp), `0x03` oder `0x85` (Tippen abbrechen) verwirft die Zeilen, die noch warten.
- **Nach einem Fehler:** Der nächste G-Code-Satz wird nicht ausgeführt und mit demselben Fehler beantwortet. Das geht so weiter, bis ein `$`-Befehl, eine leere Zeile oder ein Reset kommt.

Echtzeitbefehle

Ein Echtzeitbefehl ist ein einzelnes Byte. Die Steuerung führt ihn sofort aus, sobald er eintrifft, auch mitten in einer Zeile. Er gehört zu keiner Zeile und bekommt kein `ok`.

Byte	Befehl	Was er bewirkt
`0x18` (Strg+X)	Reset	Hält sofort alles an und löscht, was in der Warteschlange steht. Ein Reset, während sich die Achsen bewegen, löst `ALARM:3` aus: Referenzieren Sie erneut. Wird bei aktivem Not-Halt nicht angenommen.
`0x03` (Strg+C)	Reset, Not-Halt aktiv	Wie Strg+X, wird aber auch bei aktivem Not-Halt angenommen.
`0x19` (Strg+Y)	Stopp	Die Achsen bremsen bis zum Stillstand ab, und die noch eingereihten Zeilen werden verworfen. Die Position bleibt erhalten, und es wird kein Alarm ausgelöst.
`ESC`, dann `0x14`	Neustart	Startet die Steuerung neu, auf Platinen, die sich selbst neu starten können.
`?`	Statusbericht	Ein Statusbericht.
`0x80` (oder `0x05`)	Statusbericht	Ein Statusbericht, außer solange automatische Berichte eingeschaltet sind.
`0x87`	Vollständiger Statusbericht	Ein Statusbericht mit allen Feldern.
`~`, `0x81`	Zyklusstart	Setzt nach Vorschub halt, Schutztür, `M0`/`M1` oder Werkzeugwechsel fort.
`!`, `0x82`	Vorschub halt	Die Achsen bremsen bis zum Stillstand ab und halten. Bei einer Tippbewegung bricht er diese ab.
`0x83`	Parserzustand	Meldet den modalen Zustand, wie es `$G` tut.
`0x84`	Schutztür	Wirkt wie das Öffnen der Tür: anhalten und, wenn das Parken eingerichtet ist, parken.
`0x85`	Tippen abbrechen	Beendet eine Tippbewegung und verwirft die noch eingereihten Tippzeilen.
`0x88`	Wahlweiser Halt	Schaltet den wahlweisen Halt (`M1`) ein oder aus.
`0x89`	Einzelsatz	Schaltet den Einzelsatz ein oder aus.
`0x8B`	Handradmodus	Mit angeschlossenem Handrad: schaltet dessen Modus um. Sonst wird er ignoriert.
`0x8C`	Automatische Berichte	Schaltet automatische Statusberichte ein oder aus, wenn `$481` deren Intervall festlegt. Sonst wird er ignoriert.
`0x90`	Vorschub-Override 100 %	
`0x91`, `0x92`	Vorschub-Override +10 %, −10 %	Bereich 10–200 %.
`0x93`, `0x94`	Vorschub-Override +1 %, −1 %	Bereich 10–200 %.
`0x95`, `0x96`, `0x97`, `0x98`	Eilgang-Override	100 %, 50 %, 25 %, 5 %.
`0x99`	Spindel-Override 100 %	

Byte	Befehl	Was er bewirkt
`0x9A`, `0x9B`	Spindel-Override +10 %, −10 %	Bereich 10–200 %.
`0x9C`, `0x9D`	Spindel-Override +1 %, −1 %	Bereich 10–200 %.
`0x9E`	Spindelstopp	Nur bei Vorschub halt: stoppt die Spindel oder startet sie wieder.
`0x9F`	Software-Not-Halt	Hält sofort an und löst den Not-Halt-Alarm aus.
`0xA0`	Kühlung, Flutkühlung	Schaltet die Flutkühlung ein oder aus.
`0xA1`	Kühlung, Sprühkühlung	Schaltet die Sprühkühlung ein oder aus.
`0xA3`	Werkzeugwechsel fertig	Bestätigt einen manuellen Werkzeugwechsel.

Die Steuerung verwirft andere Steuerbytes und die Bytes `0x80`–`0xBF`, die keinen Befehl haben.

- **UTF-8-Text** in Kommentaren und Meldungen wird als Text gelesen. Ein Byte im Bereich `0x80`–`0xBF`, das zu einem solchen Zeichen gehört, wird nie als Echtzeitbefehl aufgefasst.
- `$39=0` (herkömmliche Echtzeitzeichen aus): `?`, `!` und `~` sind in `$`-Zeilen und Kommentaren Text und überall sonst Befehle. Mit `$39=1` (dem Standardwert) sind sie immer Befehle.

Begrüßung und Kennung

Nach dem Einschalten und nach jedem Reset meldet sich die Steuerung mit:

```
Grbl 1.1f ['$' or '$HELP' for help]
```

Sender-Programme werten eine Zeile, die mit „Grbl“ beginnt, als Neustart.

`$I` antwortet mit diesen Zeilen und dann mit `ok`:

Zeile	Was sie sagt
`[VER:Twinwright 1.1f.<date>:<text>]`	Die Protokollversion und ihr Erstellungsdatum. `<text>` ist das, was `$I=<text>` gespeichert hat.
`[OPT:…]`	Optionsbuchstaben, die Planerblöcke (100) und der Empfangspuffer (1024).
`[NEWOPT:…]`	Optionsnamen: `ENUMS`, `RT+` oder `RT-` (`$39`), `HOME`, `ES`, `MPG`, `LATHE`, `REBOOT`, `TC`, `SS` (Spindelsynchronisation), `SED`, `RTC`, `SD`, `FS`, `YM`.
`[FIRMWARE:Twinwright]`	Der Name der Firmware.
`[SIGNALS:…]`	Die Steuereingänge, die die Platine hat, als Buchstaben.
`[NVS STORAGE:…]`	Wo die Einstellungen gespeichert sind.
`[DRIVER:…]`, `[DRIVER VERSION:…]`, `[DRIVER OPTIONS:…]`, `[BOARD:…]`	Die Platine.
`[KERNEL:<version>+<build>]`	Die Version und der Build der Steuerung: der Commit eines Releases oder `dev`.
`[KNCAP:…]`	Was die Platine hat (siehe unten).
`[PLUGIN:…]`	Funktionen unter den Namen, unter denen Sender-Programme sie kennen: `MODBUS` und `HUANYANG VFD` für den Spindelantrieb.

Was die Platine hat: KNCAP

`[KNCAP:]` ist eine Zeile aus `key=value`-Paaren, durch Kommas getrennt. Schlüssel werden immer nur am Ende ergänzt und ändern nie ihre Bedeutung. Lesen Sie die Zeile nach Schlüsseln, nicht nach Positionen. Ein fehlender Schlüssel bedeutet „nicht bekannt“.

Schlüssel	Werte	Bedeutung
`port`	`teensy41`, `stm32h7`, `stm32f4`, `twin`, `host`, `sim`, `unknown`	Welcher Build der Steuerung das ist.
`pv`	eine Zahl	Die Version der Port-Schnittstelle der Steuerung.
`view`	eine Zahl	Die Version der Zustandsansicht der Steuerung.
`spindle`	`vfd`, `pwm`, `none`	Der Spindelantrieb: Modbus, ein PWM-Ausgang oder keiner.
`encoder`	`0`, `1`	Ein Spindel-Encoder ist verbaut.
`sd`	`0`, `1`	Die Platine hat einen SD-Kartenschacht.
`mpg`	`0`, `2`	Ein Handradanschluss, nach seiner Modusnummer.
`aux`	`di/do/ai/ao`	Zusatzeingänge und -ausgänge: digitale Eingänge, digitale Ausgänge, analoge Eingänge, analoge Ausgänge.

Schlüssel	Werte	Bedeutung
`axes`	Buchstaben	Die Achsen, zum Beispiel `XYZ`.
`probe`	`0`, `1`	Ein Messtastereingang. An der MC-1 nutzt ihn auch der Werkzeuglängensensor.
`link`	`usb`, `uart`, `tcp`, `none`	Die Verbindung zum Sender-Programm. `usb` ist ein serieller USB-Port, der antwortet, sobald DTR gesetzt ist.

Der Schattenmodus vergleicht `[KERNEL:]` und `[KNCAP:]` zwischen einer realen Maschine und ihrem Zwilling (siehe Echte Maschine und Schattenmodus).

Statusberichte

Ein Statusbericht kommt nach `?`, `0x80` oder `0x87`. Sind automatische Berichte eingeschaltet (`$481`, `0x8C`), kommt er auch von selbst. Er ist eine einzelne Zeile:

```
<Idle|MPos:0.000,0.000,0.000|Bf:100,1023|FS:0,0|Pn:XYZ>
```

Zuerst kommt der Zustand:

- `Idle`, `Run`, `Jog`, `Home`, `Check` (Prüfmodus), `Sleep`, `Tool` (ein Werkzeugwechsel);
- `Hold:0` (gehalten) und `Hold:1` (bremst ab);
- `Door:0` (geschlossen, bereit zum Fortsetzen), `Door:1` (angehalten, noch offen), `Door:2` (geöffnet: hält an oder parkt) und `Door:3` (geschlossen, stellt wieder her);
- `Alarm`.

Die Felder danach hängen von `$10` und von der Situation ab:

Feld	Bedeutung
`MPos:` oder `WPos:`	Maschinen- oder Werkstückposition (`$10`, Bit 0).
`Bf:`	Freie Planerblöcke, freie Bytes im Empfangspuffer.
`Ln:`	Die Satznummer des laufenden Satzes.
`FS:`	Vorschub, die angeforderte Spindeldrehzahl und die Drehzahl, die der Antrieb meldet. Eine Spindel, die nur ein- und ausgeschaltet wird, meldet `F:`. Bei Vorschub halt ist der Vorschub 0.
`Pn:`	Aktive Eingänge: Endschalter mit dem Buchstaben ihrer Achse, `P` Messtaster, `D` Tür, `H` Vorschub halt, `S` Zyklusstart, `R` Reset, `E` Not-Halt.
`WCO:`	Die Nullpunktverschiebung: alle paar Berichte und nach einer Änderung.
`Ov:`	Vorschub-, Eilgang- und Spindel-Override in %: alle paar Berichte und nach einer Änderung.
`A:`	Zubehör: `S` oder `C` Spindel im Uhrzeigersinn oder gegen den Uhrzeigersinn, `F` Flutkühlung, `M` Sprühkühlung.
`Sl:`	Die Last des Spindelantriebs.
`DTG:`	Der Restweg, bis die Achsen anhalten (`$10`, Bit 13).
`H:`	Referenziert oder nicht.
`T:`	Das Werkzeug, während eines Werkzeugwechsels.
`TLR:`	Eine Referenz-Werkzeuglänge ist gesetzt (`$TLR`).
`MPG:`	Das Handrad hat die Kontrolle (1) oder nicht (0).
`In:`	Das Ergebnis des letzten `M66`.
`WCS:`, `Sc:`, `D:`	Das Werkstückkoordinatensystem, die Skalierung, der Durchmessermodus der Drehmaschine (vollständige Berichte).
`AR`, `AR:`	Automatische Berichte aus oder ein, mit ihrem Intervall (vollständige Berichte).
`FW:`	Der Name der Firmware (vollständige Berichte).
`SD:`	Wie weit ein Programm gekommen ist, das aus dem Dateisystem der Steuerung ausgeführt wird.

Weitere Zeilen, die die Steuerung sendet

Zeile	Wann
`ALARM:N`	Ein Alarm. Alarmcodes listet jedes N auf.
`[MSG:…]`	Eine Meldung: die der Steuerung selbst oder das `(MSG,…)`, `(PRINT,…)` oder `(DEBUG,…)` eines Programms.
`[GC:…]`	Der modale Zustand, nach `$G` oder `0x83`. Mit `$10` Bit 9 einmal nach jeder Änderung.
`[PRB:x,y,z:1]`	Der Punkt, an dem der Messtaster ausgelöst hat, und ob er berührt hat (1) oder nicht (0).
`[TLO:…]`	Die Werkzeuglängenkorrektur.
`[PARAM:n=value]`	Nach `$#=n`.
`[G54:…]` … `[G59.3:…]`, `[G28:…]`, `[G30:…]`, `[G92:…]`, `[T:…]`	Nach `$#`: Nullpunktverschiebungen, gespeicherte Positionen, die Werkzeugtabelle.

Zeile	Wann
`>line:ok`, `>line:error:N`	Ein Startsatz (`$N0`, `$N1`) wurde beim Start ausgeführt.

`$`-Befehle

`$HELP Commands` listet sie auf der Steuerung selbst auf.

Befehl	Was er bewirkt
`$`, `$HELP`, `$HELP <group>`	Hilfe; die Befehle; eine Einstellungsgruppe mit Details.
`$$`, `$<n>`, `$<n>=<value>`, `$$=<n>`, `$+`	Alle Einstellungen; eine Einstellung; sie ändern; die Details einer Einstellung; noch einmal alle Einstellungen.
`$SED=<n>`	Eine Einstellung beschreiben.
`$#`, `$#=<n>`, `$#=<name>`	Nullpunktverschiebungen, das Antastergebnis, gespeicherte Positionen; ein Parameter.
`$G`	Der modale Zustand.
`$I`, `$I+`, `$I=<text>`	Was die Steuerung ist (siehe oben); alles; einen Text dazu speichern.
`$N`, `$N0=<line>`, `$N1=<line>`	Die Startsätze anzeigen oder speichern.
`$C`	Prüfmodus: Der G-Code wird geprüft, und nichts bewegt sich. Ein Reset beendet ihn.
`$X`	Nach einem Alarm entsperren.
`$H`, `$HX`, `$HY`, `$HZ`	Alle Achsen oder eine Achse referenzieren.
`$J=<G-code>`	Tippbewegung.
`$B`, `$S`, `$O`	Satzausblendung, Einzelsatz, wahlweiser Halt: ein oder aus.
`$SLP`	Ruhezustand.
`$TLR`, `$TPW`	Die letzte Antastung als Referenz-Werkzeuglänge übernehmen; das neue Werkzeug auf der Messplatte messen.
`$RST=$`, `$RST=#`, `$RST=*`	Die Werkseinstellungen der MC-1 (Referenzfahrt und Endlagen an, ihre Verfahrwege); die Nullpunktverschiebungen löschen; beides sowie die Startsätze.
`$EA`, `$EAG`, `$EE`, `$EEG`, `$EG`, `$ES`, `$ESG`, `$ESH`, `$E*`	Die Alarme, Fehlercodes, Einstellungsgruppen und Einstellungen auflisten, als Zeilen oder als CSV.
`$SPINDLES`, `$SPINDLESH`	Die Spindeln.
`$PINS`, `$PINSTATE`, `$PORTS`	Die Pins der Platine; die Zusatzeingänge und -ausgänge; die seriellen Ports.
`$LIM`, `$HSS`, `$LEV`	Die Endschaltereingänge, die Referenzschalter, die letzten Eingangsereignisse.
`$RTC`, `$RTC=<date and time>`	Die Uhr der Steuerung (ISO 8601).
`$REBOOT`	Die Steuerung neu starten.
`$F`, `$F+`, `$F=<file>`, `$F<=<file>`, `$FR`	Die Programme im Dateisystem der Steuerung auflisten, eines ausführen, eines zurücksenden, von vorn beginnen.
`$FD=`, `$FMD=`, `$FRD=`, `$FF=yes`, `$FI`, `$FM`, `$FU`, `$CWD=`, `$PWD`	Löschen, einen Ordner anlegen oder entfernen, formatieren, Informationen, einbinden, trennen, Arbeitsordner.

Jede Einstellung steht unter Einstellungen.

Dateiformate

Entwurf (Runde 14): Der Text wird noch geschrieben; die Abschnitte können noch wachsen.

Programme

Ein Programm ist eine G-Code-Textdatei im Dialekt von Grbl 1.1, den Twinwrights eigener Kernel spricht. Studio liest jede solche Datei. Die Programme, die das CAM von Studio schreibt, sehen so aus:

```
(Sensor bracket)
(tool T1 flat 6 n3 L20, 6061-T6: S13263, F836, plunge F279, depth of cut 1.20 mm, stepover 2.40 mm)
(program zero: on the blank's top; the machine's top first; T1 is the length reference)
G21 G90 G17 G94 G54
G53 G0 Z0
T1 M6
G53 G0 Z0
M3 S13263
G4 P3 (up to speed)
(face X-2..62 Y-2..42, 0.3 mm deep: 1 layer of 12 passes)
...
M5
M9
G53 G0 Z0
M30
```

Teil	Was es ist
Programmkopf	Kommentare: der Titel des Auftrags; das erste Werkzeug, das Material und die wichtigsten Schnittdaten; wo der Nullpunkt des Programms liegt
Modi	`G21 G90 G17 G94 G54`: Millimeter, absolut, die XY-Ebene, Vorschub pro Minute, Nullpunktverschiebung G54
`G53 G0 Z…`	Eine Fahrt an das obere Ende von Z, in Maschinenkoordinaten: am Anfang, bei jedem Werkzeugwechsel und am Ende. Das ist die Höhe, auf der die Referenzfahrt Z zurücklässt.
`T… M6`	Ein Werkzeugwechsel. Die Steuerung misst das neue Werkzeug am Werkzeuglängensensor.
`M3 S…`, `G4 P…`	Die Spindel ein, dann eine Verweilzeit, während sie auf Drehzahl kommt. Die Pumpe einer wassergekühlten Spindel (`M8`) kommt zuerst.
Formelement-Kommentare	Jedes Formelement beginnt mit einem Kommentar, der sagt, was es bearbeitet
Bewegungen	`G0`, `G1` und Kreisbögen `G2`/`G3` mit `I` und `J`. Gebohrt wird mit `G81` oder mit `G83` und Entspanen; die `R`-Ebene liegt 1 mm über der Oberseite.
Ende	`M5`, `M9`, die Fahrt nach oben, `M30`

Kommentare stehen in Klammern. Die Sätze haben keine Nummern.

Studio bewahrt jede Version eines Programms im Programmordner (Dokumente/Twinwright Studio/programs) auf, in einem Ordner mit dem Namen des Programms:

Datei	Was sie enthält
`v0001.nc`	Der Text des Programms, wie er gespeichert wurde
`v0001.json`	Seine Version, die Zeit (UTC), wer es erstellt hat, warum, die Version, aus der es entstand, sein Hash und seine Sätze
`v0001.ops.json`	Was aus jedem Formelement wurde (Bearbeitung, Strategie, Werkzeug, Sätze, Drehzahl, Vorschub): Das Einrichteblatt liest sie

Aufträge und Aufspannungen

Ein Auftrag ist die Beschreibung eines Teils, aus der das CAM Programme erstellt. Er ist ein JSON-Objekt:

Schlüssel	Bedeutung
`name`	Der Name des Programms, zum Beispiel `bracket.nc`
`title`, `description`	Texte für den Programmkopf und das Einrichteblatt
`material`	Das Material, zum Beispiel `6061-T6`; Bezeichnungen aus anderen Normen werden erkannt
`tool`	Das Werkzeug in der Spindel, als Spezifikation (`flat 6 n3 L20`)
`tools`	Die Werkzeugtabelle: `{"1": "flat 6 n3 L20", "2": "drill 3.4 118 L20"}`. Fehlt sie, gilt die der Maschine.
`cutting`	Werte, die die gewählten überschreiben: `rpm`, `feed`, `ap`, `ae`, `finish_ap`; `corner_steps` (true oder false: die Innenecken einer Wand zuerst in Stufen schneiden)
`safe_z`	Die Höhe, auf der die Fahrten über das Teil verlaufen, in Millimetern über dem Werkstücknullpunkt
`zero`	Wo der Werkstücknullpunkt liegt: `front-left-top` (der Standardwert) oder `center-top`
`features`	Die Formelemente, in der Reihenfolge, in der sie gefräst werden. Siehe Programme erstellen.

Schlüssel	Bedeutung
`setups`	Statt `features`: ein Objekt für jede Aufspannung
`nest`	Statt `features`: auf einer Platte verschachtelte Teile

Eine Aufspannung hat eigene `features` und gibt an, wie das Teil aus der vorhergehenden Aufspannung zu ihr gelangt:

Schlüssel	Bedeutung
`turn`	`u`: um X gewendet (vorn und hinten tauschen die Plätze); `v`: um Y (links und rechts)
`fixture`	Ab dieser Aufspannung anders gehalten (`vise`, `clamps`, `tape` …)
`zero`	Der erneut gefundene Werkstücknullpunkt: `front-left-top` oder `center-top`. Fehlt er, gilt nach einem Wenden oder einem neuen Spannmittel `front-left-top`, sonst der Nullpunkt der vorherigen Aufspannung.
`name`, `tool`, `tools`, `cutting`, `safe_z`	Die der Aufspannung selbst; sie gehen denen des Auftrags vor

```
{"name": "bracket.nc", "title": "Sensor bracket", "material": "6061-T6",
 "tools": {"1": "flat 6 n3 L20", "2": "drill 3.4 118 n2 L20"},
 "features": [
  {"op": "face", "x0": -2, "y0": -2, "x1": 62, "y1": 42, "depth": 0.3, "tool": 1},
  {"op": "pocket", "shape": "rect", "x0": 6, "y0": 6, "x1": 28, "y1": 34, "r": 3, "depth": 4, "tool": 1},
  {"op": "pocket", "shape": "circle", "cx": 44, "cy": 20, "d": "Ø12 H7", "depth": 6, "tool": 1},
  {"op": "drill", "points": [[44, 6], [44, 34]], "depth": 8, "tool": 2}]}
```

Jede Version eines Auftrags wird neben dem Programm aufbewahrt, das sie erzeugt hat, als `job-v0001.json`: mit ihrer Version, der Zeit (UTC), wer sie erstellt hat (`operator` oder `ai:<client>`), einer Notiz, der Programmversion, die sie erzeugt hat, und dem Auftrag selbst. Ein Auftrag in Aufspannungen führt außerdem für jede Aufspannung das Programm, die erzeugte Version und deren Hash auf.

Aufzeichnungen (MCAP)

`--mcap FILE` zeichnet einen Lauf in eine MCAP-Datei auf (mcap.dev, Formatversion 0): ein Kanal für jede Art von Daten, JSON-Nachrichten, jeder Kanal mit seinem JSON-Schema, nicht in Chunks aufgeteilt und nicht komprimiert, und am Ende eine Zusammenfassung mit den Statistiken und einer CRC. Die Tests von Studio lesen die Dateien wieder ein und prüfen sie gegen die MCAP-Spezifikation; Anwendungen, die MCAP lesen, wie PlotJuggler und Foxglove, sollen sie öffnen können, aber das wurde noch nicht ausprobiert.

- `--rec-period MS` legt fest, wie oft die abgetasteten Kanäle geschrieben werden (1 ms, wenn nichts angegeben ist).
- Eine Aufzeichnung ändert nichts am Lauf: Ein aufgezeichneter Lauf und derselbe Lauf ohne Aufzeichnung sind Bit für Bit gleich.
- `--play FILE.mcap` spielt eine Aufzeichnung in Studio ab, mit einer Zeitleiste: Abspielen, Pausieren, Springen, 0,25- bis 10-fache Geschwindigkeit. Während des Abspielens wird nichts simuliert.

Zeit

Die Log-Zeit jeder Nachricht ist die virtuelle Zeit des Zwillings, in Nanosekunden ab dem Einschalten. Die meisten Kanäle tragen außerdem `t`, dieselbe Zeit in Sekunden. Ein abgetasteter Kanal schreibt nur dann eine Nachricht, wenn sich etwas darin geändert hat.

Der Dateikopf

Ein Metadatensatz mit dem Namen `mc1sim` sagt, was der Lauf war:

Schlüssel	Bedeutung
`machine`	Der Name der Maschine
`drive_model`	Das Antriebsmodell des Laufs: `L0`, `L1`, `L2`, `L2/3` (drei Massen) oder `L3`
`f_cpu`	Der Takt der Steuerung, Hz
`nvs`	Ein Hash der gespeicherten Einstellungen der Steuerung zu Beginn des Laufs, 16 Hexadezimalziffern
`sample_period_ms`	Die Abtastperiode
`time`	`virtual, ns from power-on`

Kanäle

Kanal	Geschrieben	Feld	Typ	Einheit	Bedeutung
`/machine`	abgetastet	`pos`	number[3]	mm	Wo jede Achse steht: die Mechanik, nicht die Vorstellung der Steuerung davon
		`mpos`	number[3] oder null	mm	Die Maschinenposition der Steuerung (MPos); null, solange die Steuerung aus ist
		`lost`	number[3]		Die Schritte, die jede Achse verloren hat
		`feed`	number	mm/min	Der aktuelle Vorschub der Steuerung

Kanal	Geschrieben	Feld	Typ	Einheit	Bedeutung
		`tip`	number[3]	mm	Die Werkzeugspitze gegenüber dem Werkstück: bei eingeschaltetem Fehlermodell des Zwillings um dessen Fehler verschoben, bei ausgeschaltetem dasselbe wie `pos`
`/drives`	abgetastet, mit einem Antriebsmodell ab L1	`ferr_um`	number[3]	µm	Der Schleppfehler jedes Antriebs
		`state`	string[3]		Der Zustand jedes Antriebs
		`km1`	boolean		Das Schütz KM1 ist geschlossen
		`bus48`	boolean		Die 48-V-Versorgung der Antriebe ist eingeschaltet
`/spindle`	abgetastet, mit einem Spindelantrieb	`f_out`	number	Hz	Die Ausgangsfrequenz des Antriebs
		`rpm`	number	min^{-1}	Die Drehzahl der Spindel
		`amps`	number	A	Der Ausgangsstrom des Antriebs
		`load_pct`	number	%	Die Last, wie der Antrieb sie meldet
		`torque_nm`	number	N·m	Das Drehmoment des Motors
		`bus_v`	number	V	Die Zwischenkreisspannung des Antriebs
		`cmd_rpm`	number	min^{-1}	Die Drehzahl, die die Steuerung vorgibt
		`run`, `pump`	boolean		Der Antrieb läuft; die Pumpe ist an
		`power`	string		Die eigene Versorgung des Antriebs: `ready`, `boot` oder `off`
		`fault`	string		Der Fehler des Antriebs, wie ihn dessen Anzeige zeigt
`/controller`	bei Änderung	`state`	string		Der Zustand der Steuerung (`Off`, solange sie aus ist)
		`alarm`	integer		Der Alarm, in dem sie sich befindet (0: keiner)
		`homed`	integer		Die referenzierten Achsen, je ein Bit: X 1, Y 2, Z 4
		`limits`	integer		Die aktiven Endschaltereingänge, je ein Bit
		`probe`, `estop`, `motor_fault`, `door`	boolean		Die Eingänge für Messtaster, Not-Halt, Motorfehler und Tür (die MC-1 hat keine Tür)
		`on`	boolean		Die Steuerung ist eingeschaltet
`/power`	bei Änderung	`mains`, `v24`, `mcu`, `km1`, `bus48`	boolean		Netz; die 24-V-Versorgung; die laufende Steuerung; KM1; die 48-V-Versorgung
		`vfd_mains`, `vfd`, `pump`	boolean, string, boolean		Das Netz des Spindelantriebs; sein Zustand (`none`, `ready`, `boot`, `off`); die Pumpe
`/serial/in`	sobald es eintrifft	`text`	string		Die Bytes, die die Steuerung empfangen hat; Steuerzeichen als `\u`-Escapes
`/serial/out`	sobald es eintrifft	`line`	string		Jede Zeile, die die Steuerung geschrieben hat
`/events`	sobald sie auftreten	`text`	string		Jeder Eintrag des Ereignisprotokolls: Marken, die Versorgung, die Antriebe, der Spindelantrieb, Kollisionen, Kabelbrüche
`/cut`	abgetastet, mit einem Rohteil	`cutting`	boolean		Es wird Material abgetragen; solange nicht, sind die folgenden Werte 0, außer `rpm` und `removed_mm3`
		`mrr_mm3_s`	number	mm^3/s	Die Materialabtragsrate
		`ap_mm`, `ae_mm`	number	mm	Die axiale Schnitttiefe und die radiale Schnittbreite
		`fz_mm`	number	mm	Der Vorschub pro Zahn
		`rpm`	number	min^{-1}	Die Drehzahl der Spindel
		`fx_n`, `fy_n`, `fz_n`	number	N	Die Schnittkraft entlang der Maschinenachsen
		`torque_nm`, `power_w`	number	N·m, W	Das Schnittmoment und die Schnittleistung
		`kc_j_mm3`	number	J/mm^3	Die spezifische Schnittenergie
		`removed_mm3`	number	mm^3	Das bisher abgetragene Material
`/input`	sobald es eintrifft	`line`	string		Die Eingaben der Sitzung, Zeile für Zeile, wie `--record` sie schreibt (siehe Sitzungsprotokolle)
`/shadow`	sobald es eintrifft, im Schattenmodus	`host_s`	number	s	Die Zeit des Computers
		`dir`	string		`t` zur realen Maschine, `r` von ihr
		`data`	string		Die Bytes, jedes als ein Zeichen U+0000–U+00FF, bis zu 1 KB pro Nachricht

Kanal	Geschrieben	Feld	Typ	Einheit	Bedeutung
`/ui/state`, `/sound`	für den Player				Der Zustands-Frame der Ansichten (30 pro Sekunde, solange er sich ändert) und der Frame des Klangs (alle 10 ms virtueller Zeit, wenn er sich ändert). Eigene Kanäle von Studio: Ihre Felder können sich von Version zu Version ändern.

Pin-Verläufe (VCD)

`--vcd FILE` schreibt die Pins der Steuerung als VCD-Datei (IEEE 1364), die GTKWave und PulseView lesen: Der Scope `mcu` enthält die Pins der Steuerung, jede Änderung zu dem Taktschritt, in dem sie geschieht (die Schritt-Pins als Impulse von `$0` Mikrosekunden); der Scope `machine` enthält die Leitungen außerhalb der Steuerung: die Versorgungen, KM1, den Spindelantrieb, die Pumpe, den Not-Halt-Taster, den Werkzeuglängensensor, die fünf Schalter. Die Zeitskala beträgt 1 ns.

Sitzungsprotokolle

`--record FILE` schreibt die Eingaben einer Sitzung, jede mit ihrer virtuellen Zeit: was die Sender-Programme gesendet haben, die Echtzeitbefehle, die Befehle der Ansichten. `--replay FILE` führt die Sitzung Bit für Bit noch einmal aus, mit derselben Version von Studio und denselben Maschinendateien. `--replay FILE --branch S` spielt sie bis S Sekunden ab und läuft von dort live weiter; mit `--record` hat der Zweig ein eigenes Protokoll, das vom übergeordneten Protokoll ausgeht (`# branch <parent> <ticks>`). Ein Protokoll ist Text; seine Zeilen sind Studios eigenes Format und können sich von Version zu Version ändern.

Das Weltarchiv

Was der Zwilling zwischen den Sitzungen behält, so wie es eine reale Maschine tut, wenn sie aus- und wieder eingeschaltet wird: wo die Achsen stehen, das Rohteil mit seinen Schnitten, das Werkzeug in der Spindel, der Speicher der Steuerung und die Uhr der Platine sowie die Wärme, die abkühlt, während die Maschine aus ist. Die Steuerung startet kalt: Die Maschine wird erneut referenziert.

Das Fenster bewahrt die Welt jeder Maschine im Zustandsordner auf (unter Windows `%LOCALAPPDATA%\Twinwright\Studio`): die der ersten Maschine in `world`, die der anderen in `machines\<id>\world`. `--world DIR` verwendet einen anderen Ordner; `--world-reset` beginnt mit dem Werkszustand. Stapelläufe, Tests, Wiederholungen, abgespielte Aufzeichnungen und die Hintergrundmaschinen der KI behalten keine Welt, es sei denn, `--world` ist angegeben. Eine Instanz zur Zeit öffnet eine Welt; eine zweite beginnt mit dem Werkszustand und speichert nichts.

Datei	Was sie enthält
`world.json`	Der Stand am Ende der Sitzung (siehe unten), zuerst nach `world.json.tmp` geschrieben, dann umbenannt
`stock.snap`	Die Zellen des Rohteils, geschnitten oder nicht (binär, `TWSTOCK1`)
`eeprom.bin`	Der nichtflüchtige Speicher der Steuerung: ihre Einstellungen, Nullpunktverschiebungen und Werkzeugdaten, bei jeder Änderung sofort durchgeschrieben

`world.json`:

Schlüssel	Bedeutung
`world`	Die Version des Formats: 1
`saved_at`, `saved_epoch`	Wann es gespeichert wurde: UTC und Sekunden seit 1970
`version`	Die Version von Studio, die es gespeichert hat
`pos`	Wo die Achsen standen, mm (X, Y, Z)
`tool`	Das Werkzeug in der Spindel, als Spezifikation (`flat 6 n3 L20`)
`tool_table`	Die Werkzeugtabelle des Auftrags: `{"1": "flat 6 n3 L20", ...}`
`tool_auto`	Der Zwilling spielt bei einem Werkzeugwechsel den Bediener: Er setzt das Werkzeug aus der Tabelle ein und drückt Zyklusstart
`setup`	Die Werkstückspannung und das Rohteil, wie sie „Aufspannung speichern" im Fenster schreibt (null: keine)
`cells`	`stock.snap` enthält die Zellen des Rohteils
`thermal`	Die Wärme: `screws` (jede Gewindespindel über dem Rahmen), `frame` (über 20 °C), `ambient` (der Raum über 20 °C), `part_heat` und `part_offset` (das Werkstück über dem Rahmen: was die Schnitte einbringen, und ein vorgegebener Versatz), alle in K; `spindle_um`, die Längung der Spindel in µm
`rtc`	Der Stand der Uhr der Platine, s
`probe_cal`	Der Messtaster, wie er zuletzt kalibriert wurde: der Durchmesser der Kugel, ihr wirksamer Radius, das Lobing und seine Phase, der Durchmesser der Ringlehre, die größte Abweichung, wann und von wem (erst nach einer Kalibrierung)
`odometer`	`travel_mm` und `reversals` jeder Achse, `spindle_h` (die Stunden der Spindel) und `on_h` (die Stunden der Anwendung), über die Sitzungen aufsummiert

Beim nächsten Start stehen die Achsen dort, wo sie standen; dasselbe Rohteil kommt mit seinen Zellen zurück, ebenso das Werkzeug und die Tabelle; die Wärme hat sich so lange abgekühlt, wie die Maschine aus war, jedes Teil mit seiner eigenen Zeitkonstante seinem Ruhewert entgegen; die Uhr der Platine ist weitergelaufen. `--world-off S` gibt an, wie lange die Maschine aus war (für Tests; sonst die Zeit seit dem Speichern). Beim Start angegebene `--pos`, `--tool`, `--stock`, `--setup` und `--part` haben Vorrang vor dem Archiv.

Lizenzdateien

Eine Lizenzdatei (`.twlic`) ist ein JSON-Objekt in UTF-8, signiert mit Ed25519. Studio prüft sie auf dem Computer, ohne Netzwerk. Wo die Dateien liegen und wie sie verwendet werden: Lizenzen, Arbeitsplätze und Aktivierung.

Dieses Beispiel stammt aus den gemeinsamen Testvektoren. Es ist mit dem Testschlüssel signiert (`kid` `test`), den Release-Builds nicht akzeptieren:

```
{"v": 1, "kid": "test", "id": "TWL-TEST000001", "edition": "pro", "kind": "subscription", "seats": 1,
 "holder": "Weber Lab, TU Example", "issued": "2026-11-02T10:00:00Z", "valid_until": "2027-11-02T10:00:00Z",
 "renew_after": "2027-10-28T10:00:00Z", "updates_until": null, "fp": ["0123456789abcdef0123456789abcdef"],
 "mc1_serial": null,
 "sig": "h3FTqZnDYtq8Q7_0ZuELLeDrTfyZnkTPgvDWSywTyl7BCJJxGIyhrBQUjYHJiE1_WOWaFarqbveFiMLrEfUyCA"}
```

Felder

Feld	Typ	Bedeutung
`v`	integer	Die Version des Formats: 1. Studio behandelt eine Datei einer anderen Version als ungültig.
`kid`	string	Welcher der Schlüssel von Twinwright die Datei signiert hat
`id`	string	Die Nummer der Lizenz: `TWL-` und 10 Buchstaben und Ziffern. Die Verlängerung sendet sie.
`edition`	string	`maker` oder `pro`
`kind`	string	`subscription`, `perpetual` (Kauflizenz) oder `owner` (MC-1-Besitzer)
`seats`	integer	Die Anzahl der Arbeitsplätze
`holder`	string	Für wen die Lizenz gilt: eine Person oder eine Organisation. Nie eine E-Mail-Adresse.
`issued`	Datum und Uhrzeit	Wann die Datei signiert wurde, UTC, `YYYY-MM-DDTHH:MM:SSZ`
`valid_until`	Datum und Uhrzeit oder null	Abonnement: das Ende des bezahlten Zeitraums. Kauflizenz: das Ende des Erstattungszeitraums in der ersten Datei, null in der dauerhaften.
`renew_after`	Datum und Uhrzeit oder null	Ab wann Studio versucht, die Datei zu verlängern
`updates_until`	Datum oder null	Kauflizenz, MC-1-Besitzer: Die bis zu diesem Tag veröffentlichten Versionen haben die kostenpflichtigen Funktionen
`fp`	Array von Zeichenketten	Die Computer, an die die Lizenz gebunden ist: die Hashes ihrer Fingerabdrücke, bis zu drei je Arbeitsplatz
`mc1_serial`	string oder null	MC-1-Besitzer: die Seriennummer der Maschine. Studio prüft die Maschine beim Betrieb nicht.
`sig`	string	Die Ed25519-Signatur der anderen Felder, base64url ohne Padding

Ein Feld, das Studio nicht kennt, ist Teil der signierten Bytes, und Studio ignoriert seine Bedeutung.

Die signierten Bytes

Die Signatur deckt die Datei ohne `sig` ab, geschrieben in einer kanonischen Form:

- Schlüssel in der Reihenfolge der Unicode-Codepoints;
- kein Leerraum;
- Zeichenketten mit den kürzesten Escape-Folgen von JSON;
- Ganzzahlen ohne Dezimalpunkt; keine anderen Zahlen, und kein Schlüssel doppelt.

Studio schreibt die Datei in dieser Form, bevor es die Signatur prüft. Leerzeichen, Zeilenumbrüche und die Reihenfolge der Schlüssel in der Datei spielen deshalb keine Rolle. Ein geänderter Wert lässt die Prüfung fehlschlagen.

Der Hash des Fingerabdrucks

Der Hash besteht aus den ersten 32 Hexadezimalziffern von SHA-256 über die Bytes `twinwright-fp-v1`, ein Nullbyte und die ID des Computers:

System	Die ID des Computers
Windows	Der Wert `MachineGuid` unter `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography`
Linux	`/etc/machine-id`

Ohne beides erzeugt Studio eine zufällige ID und legt sie im Lizenzordner ab (`machine.id`); sie zu löschen macht den Computer zu einem anderen. Die ID selbst verlässt den Computer nie.

Zum Beispiel ergibt die ID `8b1c3a0e-5f2d-4c7a-9e61-2d0f4b7a1c55` den Hash `0a2d31bb82176516415efa7b3a340562`.

Der Anforderungscode

Der Anforderungscode für die Offline-Aktivierung ist das kanonische UTF-8-JSON `{"fp": <the hash>, "version": <Studio's version>}` in base64url ohne `=`-Padding. Zum Beispiel ergeben der Hash `0123456789abcdef0123456789abcdef` und die Version 1.0.0:

```
eyJmcCI6IjAxMjM0NTY3ODlhYmNkZWYwMTIzNDU2Nzg5YWJjZGVmIiwidmVyc2lvbiI6IjEuMC4wIn0
```

`TwinwrightStudio.exe --licence-request` gibt den Anforderungscode dieses Computers aus.

Testdateien

Eine Testphase ist eine Lizenzdatei wie die anderen, vom Lizenzserver von Twinwright signiert: `"kind": "trial"`, der Hash dieses Computers in `fp` und `valid_until` 30 Tage nach Beginn der Testphase. Sie wird nie verlängert. Eine von Hand geänderte Testdatei gilt nicht mehr.

Das Maschinenarchiv

Jeder Schlüssel des Archivs einer Maschine, mit den Werten der MC-1: Schema des Maschinenprofils.

Sender-Software mit Twinwrights eigenem Kernel

Diese Seite ist für alle, die die MC-1 mit einem eigenen G-Code-Sender-Programm (bCNC, gSender oder Universal G-code Sender) statt mit Studio ansteuern wollen. Die Steuerung der MC-1 ist Twinwrights eigener Kernel, kompatibel mit dem seriellen Protokoll von Grbl 1.1 und den Erweiterungen, die aktuelle Sender-Programme verwenden; diese Sender-Programme funktionieren daher damit, mit den wenigen Einstellungen und Vorbehalten weiter unten.

Alles hier wurde am digitalen Zwilling der MC-1 getestet, der denselben Steuerungscode wie die Maschine ausführt. Die Prüfung an einer realen Maschine steht noch aus. Getestete Versionen: bCNC 0.9.16, gSender 1.6.4, UGS Platform 2.1.27.

Verbinden

Wo	Wie
Der Zwilling im Fenster von Studio	Studio akzeptiert jeweils ein Sender-Programm auf `localhost:2323` (nur dieser Computer; starten Sie Studio mit `--bind any`, um es aus dem Netzwerk zu erreichen). bCNC: Port `socket://localhost:2323`. gSender: Ethernet, Adresse `127.0.0.1`, Port `2323`. UGS: `tcp://localhost:2323`.
Die MC-1 selbst	Natives USB: Die Steuerung erscheint als serieller Port (COM*n* unter Windows). Wählen Sie diesen Port. Die Baudrate spielt bei nativem USB keine Rolle, aber einige Sender-Programme verlangen eine: 115200 funktioniert.

Was die Steuerung über sich selbst sagt:

- Ihre Begrüßungszeile lautet `Grbl 1.1f ['$' or '$HELP' for help]`, wie bei einer klassischen Grbl-Steuerung. Sender-Programme suchen nach einer Begrüßungszeile, die mit „Grbl" beginnt, um zu erkennen, dass die Steuerung neu gestartet ist (siehe Abschnitt 3).
- Ihr Name steht in der Antwort auf `$I`: `[VER:Twinwright 1.1f.…]` und `[FIRMWARE:Twinwright]`. Sender-Programme, die die Firmware anzeigen, zeigen „Twinwright".

Die Firmware im Sender-Programm wählen

Sender-Programm	Einstellung	Warum
bCNC 0.9.16	Controller: **GRBL1**	bCNC liest den Namen der Firmware nicht. Mit GRBL1 getestet.
gSender 1.6.4	Default firmware: die andere Auswahl (nicht **Grbl**)	gSender erkennt eine Firmware an ihrem Namen in `$I`. Unsere Steuerung nennt ihren eigenen Namen, daher verwendet gSender die Firmware, die Sie als Standard eingestellt haben. Mit dieser Auswahl getestet.
UGS 2.1.27	Firmware: **GRBL**	Getestet: Einstellungen, Programme, Alarme und Referenzfahrt nach einem Reset, ohne zusätzlichen Soft Reset (Abschnitt 3). Die andere Firmware-Auswahl von UGS funktioniert ebenfalls, braucht aber nach einem Reset einen weiteren Soft Reset; der Tippbetrieb wurde nur mit dieser Auswahl getestet.

Nach einem Alarm oder einem Reset

- **Not-Halt.** Der Not-Halt der MC-1 öffnet das Leistungsschütz. Entriegeln Sie ihn, drücken Sie die Start-Taste der Maschine, führen Sie dann im Sender-Programm einen Reset aus und referenzieren Sie. Solange Start nicht gedrückt wird, lässt sich der Alarm (ALARM:10) nicht aufheben.
- **ALARM:11 (Referenzfahrt erforderlich).** Referenzieren Sie (`$H`). Das Entsperren mit `$X` wird abgelehnt (error:46).
- **ALARM:2 (Software-Endschalter).** Das Programm hat eine Bewegung außerhalb des Verfahrwegs verlangt. Die Referenzfahrt lässt Z am oberen Ende seines Verfahrwegs stehen (Maschinen-Z 0); daher löst jede Bewegung darüber, etwa `G53 G0 Z5`, ALARM:2 aus. Führen Sie einen Reset aus und referenzieren Sie dann oder entsperren Sie.
- **bCNC** erholt sich nach jedem Reset: Es sieht die Begrüßungszeile „Grbl" und vergisst die Zeilen, die noch unterwegs waren, sodass `$H` abgeschickt wird.
- **bCNC nach Stop.** Stop in bCNC ist ein Vorschub halt, dann ein Reset. Danach schickt bCNC die Modi, die es vorher hatte (Koordinatensystem, Einheiten, Werkzeuglängenkorrektur), in ein oder zwei Zeilen an die Steuerung zurück, und in einigen Fällen lehnt die Steuerung sie ab, wie es andere Grbl-Steuerungen tun: Ist eine Werkzeuglängenkorrektur (`G43.1`) wirksam, bekommt der Satz error:24 (bCNC sendet nach seinem ersten Stop ein eigenes `G43.1Z...`, sodass es ab dem zweiten Stop immer passiert); unter `G80` bekommt `G43.1Z...` error:31; ist eine Skalierung (`G51`) wirksam, bekommt `G51:X` error:1. Nach einem Fehler bekommt jeder folgende G-Code-Satz denselben Fehler, bis eine leere Zeile, ein `$`-Befehl oder ein Reset kommt. Prüfen Sie daher nach einem Stop das Koordinatensystem, die Einheiten und die Werkzeuglängenkorrektur mit `$G` und `$#`, bevor Sie im Tippbetrieb verfahren oder weitermachen. Sind sie falsch, setzen Sie den Nullpunkt erneut und messen Sie die Werkzeuglänge erneut (am Zwilling getestet, 2026-10-06).
- **UGS mit seiner anderen Firmware-Auswahl.** Nach einem Reset, den UGS nicht selbst gesendet hat, zum Beispiel einem aus dem Fenster von Studio oder bei einem Neustart der Steuerung von selbst, drücken Sie in UGS **noch einmal Soft Reset**, bevor Sie Home drücken. Bis dahin tut Home nichts. Mit dieser Auswahl wertet UGS nur eine andere Begrüßungszeile als Neustart. Ein Reset über die eigene Soft-Reset-Schaltfläche von UGS ist unproblematisch. An der MC-1 selbst sind solche Resets selten: Sie hat keinen Reset-Eingang und einen USB-Host. Ist die Firmware auf „GRBL" eingestellt, tritt das nicht auf: Nach einem solchen Reset zeigt UGS CONNECTING, und Home funktioniert sofort (am Zwilling getestet, 2026-10-06).

Werkzeugwechsel (M6)

Keines der drei Sender-Programme lässt die MC-1 mit den Standardeinstellungen das Werkzeug wechseln und das neue messen:

Sender-Programm	Standard	Was zu tun ist
bCNC	Überspringt jeden M6-Satz: Die Schnitte des nächsten Werkzeugs werden mit dem Werkzeug gemacht, das noch in der Spindel steckt.	Stellen Sie seinen Werkzeugwechsel auf **Manual tool change (TLO)** und messen Sie am Werkzeuglängensensor der MC-1 (Abschnitt 5).

Sender-Programm	Standard	Was zu tun ist
gSender	Werkzeugwechsel „Ignore“: M6 wird als Kommentar gesendet, und das Programm hält nicht an.	Stellen Sie seinen Werkzeugwechsel auf **Fixed Tool Sensor** am Werkzeuglängensensor der MC-1 (Abschnitt 5).
UGS	Behandelt M6 selbst: Es fährt zurück, stoppt die Spindel und fordert Sie auf, das Werkzeug zu wechseln, misst es aber nie. Ein längeres Werkzeug schneidet dann zu tief, und an einer realen Maschine kann das Werkzeug mit dem Werkstück kollidieren.	Führen Sie ein Werkzeug pro Programm aus und setzen Sie vor jedem die Länge des Werkzeugs oder den Werkstücknullpunkt, oder verwenden Sie Studio für Programme mit Werkzeugwechseln.

Mehr zu UGS und M6:

- Es hält auch bei einem „M6“ in einem Kommentar an, etwa bei `( M6 T1 )` von FreeCAD.
- Ein zweites M6, das direkt nach einem Werkzeugwechsel kommt, kann an ihm vorbei zur Steuerung gelangen. Die Steuerung wartet dann im Zustand Tool auf eine Bestätigung, die UGS nie sendet, und keine Schaltfläche im Fenster von UGS funktioniert mehr. An der MC-1 beendet nur der Not-Halt das: Drücken Sie ihn, entriegeln Sie ihn, drücken Sie Start, dann Soft Reset und Home in UGS.
- Wird die Steuerung während eines UGS-Auftrags zurückgesetzt, bevor UGS ein M6 erreicht hat, meldet UGS den Auftrag unter Umständen als abgeschlossen, aber sein Werkzeugwechsel-Dialog erscheint trotzdem. Wenn Sie ihn schließen, kann das ganze Programm noch einmal von vorn beginnen. Schließen Sie nach einem Reset mitten in einem Auftrag die Datei in UGS, bevor Sie weitermachen.

Die Werkzeuglänge am Werkzeuglängensensor der MC-1 messen

Der Werkzeuglängensensor sitzt hinten links am Tisch; `G59.3` ist direkt darüber gespeichert.

- **bCNC, Manual tool change (TLO).** Werkzeugwechselposition über dem Werkzeuglängensensor. Antastposition `G59.3`. Antastweg 50 mm, Suchgeschwindigkeit 200 mm/min, Feinvorschub 25 mm/min (`$342`–`$344` der MC-1), Befehl G38.2. Messen Sie das erste Werkzeug mit „Calibrate“, bevor Sie das Programm ausführen. bCNC setzt dann für jedes neue Werkzeug `G43.1`.
- **gSender, Fixed Tool Sensor.** Sensorposition `G59.3`. Messen Sie das Anfangswerkzeug im Assistenten. gSender verschiebt dann für jedes neue Werkzeug das Werkstück-Z (`G10 L20`), statt eine Werkzeuglängenkorrektur zu verwenden. Es nimmt die Länge von der Stelle, an der die Achse nach der Berührung stehen geblieben ist, nicht vom Auslösepunkt, den die Steuerung meldet; jeder Wechsel kann daher etwas abweichen. Fahren Sie im Zwilling Werkzeugwechsel mit 1× Geschwindigkeit: gSender liest diese Position nach einer festen kurzen Pause, und ein schneller laufender Zwilling kann ihm eine Position von vor dem Stillstand liefern.
- **Beginnen Sie mindestens 0,1 mm unterhalb des oberen Endes von Z** (Maschinen-Z −0,1 oder tiefer), wenn der Werkzeugwechsel beginnt. Beide Sender-Programme fahren nach dem Messen auf diese Höhe zurück. Wenn Sie ganz oben beginnen, kann die kleine Streuung einer Messung diese Rückfahrt über den Software-Endschalter hinaus führen, und der Auftrag hält mit ALARM:2 an, bevor er schneidet.

Den Werkstücknullpunkt mit einem Messtaster setzen

- **UGS** (sein Antastmodul) berechnet den Nullpunkt aus den Auslösepunkten, die die Steuerung meldet (`[PRB:]`). Das ist der richtige Weg.
- **gSender** (3D-Messtaster) und **bCNC** (seine Messtaster-Seite, dann seine Schaltflächen Z=0 / X / Y) setzen den Nullpunkt dort, wo die Achsen nach der Berührung stehen geblieben sind. Die Maschine läuft nach dem Auslösen immer ein Stück weiter, bevor sie anhält, und je schneller die letzte Berührung, desto weiter. Verwenden Sie eine langsame letzte Berührung, oder setzen Sie den Nullpunkt mit den Antastzyklen von Studio, die die Auslösepunkte verwenden.
- **Die Lochmitte von UGS** (die Funktion Hole Center seines Antastmoduls): Von dort, wo Sie den Messtaster stehen lassen, tastet es nach -X, +X, -Y und +Y, jeweils um den halben eingegebenen Lochdurchmesser, und setzt die Mitte genau zwischen die Auslösepunkte der langsamen Berührungen. Wie weit es reicht, hängt von diesem Durchmesser ab: Der Start darf höchstens (eingegebener Durchmesser − tatsächliche Bohrung) / 2 + Radius der Kugel von der Mitte entfernt sein. Wird der Durchmesser der Bohrung selbst eingegeben, ist das nur der Radius der Kugel (1 mm bei einer 2-mm-Kugel); geben Sie ihn einige mm größer ein (24 bis 26 bei einer 20-mm-Bohrung), um Spielraum zu lassen, wenn Sie den Messtaster nach Augenmaß platzieren.
- **Wenn die Lochmitte von UGS die Wand nicht erreicht** (Befund 192, Verhalten von UGS selbst): Dieser Antastvorgang endet mit ALARM:5, und UGS hält an. Es hat aber vor dem Antasten X0 Y0 von G54 am Startpunkt gesetzt, und da der Zyklus nicht beendet wurde, bleibt G54 dort. Setzen Sie nach dem Entsperren den Nullpunkt erneut, bevor Sie schneiden. Sein Antastzyklus bleibt außerdem auf halbem Weg stehen: Trennen Sie die Verbindung und verbinden Sie sich erneut, bevor Sie wieder antasten. Es hält außerdem einen Punkt, den die Steuerung als nicht berührt meldet (`[PRB:...:0]`), für eine Berührung.
- Kalibrieren Sie den Messtaster zuerst (Antastzyklen von Studio: eine Ringlehre). Ein nicht kalibrierter Messtaster misst jede Bohrung um den doppelten Vorlaufweg zu groß und jeden Zapfen um den doppelten Vorlaufweg zu klein.

Weitere Hinweise

- **bCNC über das Netzwerk.** Unter Windows liest bCNC eine TCP-Verbindung nur etwa alle 2 Sekunden, sodass seine Anzeige 2 Sekunden alt sein kann. „Idle“ in bCNC bedeutet nicht, dass die Maschine angehalten hat. Eine USB-Verbindung hat diese Verzögerung nicht.
- **gSender, M0.** Ein M0 in den ersten 10 Sätzen eines Programms pausiert nicht (gSender macht es zu einem Kommentar). M1 pausiert immer.
- **gSender, kontinuierlicher Tippbetrieb.** Das Gedrückthalten einer Tipp-Schaltfläche wird an der MC-1 abgelehnt (error:15): gSender verlangt eine lange Tippbewegung über den Verfahrweg hinaus. Verfahren Sie in Schritten. Bei manchen Grbl-Steuerungen beendet `$40=1` (Tippbefehle begrenzen) lange Tippbewegungen an den Software-Endschaltern, statt sie abzulehnen; die Steuerung der MC-1 wertet diese Einstellung noch nicht aus und lehnt sie weiterhin ab (mit gSender am Zwilling ausprobiert, 2026-10-06).
- **UGS, lange Sätze.** UGS sendet nie einen Satz, der länger als 127 Zeichen ist: Der Auftrag hält ohne Meldung an. Halten Sie die Sätze kürzer (UGS entfernt zuerst die Kommentare).
- **UGS, M30.** UGS entfernt M30. Ein Programm, das sich darauf verlässt, dass M30 Spindel und Kühlung stoppt, lässt sie laufen: Beenden Sie Programme mit M5 und M9 vor M30.
- **UGS, Kommentare.** Hat ein Satz zwei Kommentare, entfernt UGS auch den Code dazwischen.

- **Trennen der Verbindung während eines Auftrags.** Die Steuerung führt die Sätze, die sie schon hat, weiter aus. gSender und UGS senden beim Trennen kein Stopp, sodass die Maschine weiterläuft, mit drehender Spindel. Beenden Sie den Auftrag, bevor Sie die Verbindung trennen. bCNC hält an und setzt zurück, bevor es die Verbindung schließt.
- **Kommentare mit Nicht-ASCII-Zeichen** (etwa Chinesisch, „Ø“, „–“): Alle drei Sender-Programme entfernen Kommentare, bevor sie senden, sodass diese Zeichen die Steuerung nie erreichen. Die Steuerung selbst liest UTF-8-Text in Kommentaren und in `(MSG,...)` als Text, nie als Echtzeitbefehl.