Twinwright Studio 1.2.1

Twinwright Studio manual

2026-10-16

Índice

User manual

1. Primeros pasos
2. La ventana y los espacios de trabajo
3. Máquinas y encendido
4. Preparar un trabajo
5. Palpar y buscar el origen de pieza
6. Crear programas (CAM)
7. Programas y el editor
8. Ejecutar un programa y reanudarlo
9. Análisis y diagnóstico
10. Ejecutar un programa en el gemelo
11. Inspección, informes y hojas de amarre
12. Conectar una máquina real
13. El asistente de IA
14. Idioma y unidades
15. Teclado y accesibilidad
16. Solución de problemas

Operations manual

1. Requisitos del sistema
2. Instalar y desinstalar
3. Licencias, puestos y activación
4. Sus datos y copias de seguridad
5. Actualizar Studio
6. Privacidad y lo que envía Studio
7. Laboratorios, aulas y equipos compartidos

Technical manual

1. Los modelos del gemelo y sus límites
2. El núcleo propio de Twinwright
3. Código G y código M
4. Estrategias y parámetros de CAM
5. Ajustes, descentramiento y métodos de inspección
6. Perfiles de máquina
7. Hardware del MC-1

Reference

1. MCP tool reference
2. Referencia de ajustes
3. Códigos de alarma
4. Códigos de error
5. Command line
6. Machine profile schema
7. Protocolo
8. Formatos de archivo
9. Programas de envío con el núcleo propio de Twinwright

User manual

Primeros pasos

Twinwright Studio ejecuta un gemelo digital de su máquina CNC. Usted prepara la pieza, crea o abre un programa y lo ejecuta en el gemelo antes de que la máquina real corte. Este capítulo le guía desde la instalación hasta una primera pieza en unos diez minutos.

Instalación

Studio necesita Windows 10 u 11 de 64 bits (vea Requisitos del sistema).

1. Ejecute `TwinwrightStudio-<version>-setup.exe`. No necesita permisos de administrador: Studio se instala para usted.
2. Lea el contrato de licencia y haga clic en **I Agree**.
3. Conserve la carpeta o elija otra.
4. En la última página, deje marcada la casilla **Run Twinwright Studio** y haga clic en **Finish**.

Studio está en el menú Inicio como **Twinwright Studio**. La instalación en muchos equipos y qué va en cada lugar: Instalar y desinstalar.

El primer arranque

Studio se abre en el espacio de trabajo **Ejecución** con la máquina **MC-1 · Mi banco de trabajo** encendida. No se pregunta nada.

La barra de la máquina muestra **ALARM:11** y **Siguiente paso: Referencia**. Esto es normal: como una máquina real, el gemelo no sabe dónde están sus ejes hasta que hace la referencia.

Lo que muestra la ventana

Una instalación nueva muestra todo lo que usa un ingeniero: los seis espacios de trabajo, las curvas y las coloraciones. Si solo ejecuta programas o enseña, elija un nivel que muestre menos: **Herramientas > Nivel de interfaz** (vea Niveles de usuario).

Idioma y unidades

Studio arranca en el idioma de su sistema cuando lo tiene (inglés, chino, francés, alemán, japonés o español) y, si no, en inglés. Las unidades y el aspecto de los números siguen su región.

Para cambiarlos: **Herramientas > Language / 语言** y **Herramientas > Unidades**. Vea Idioma y unidades.

Ediciones y la prueba

La edición que tiene se muestra en la barra superior, por ejemplo **Learning**. Haga clic en ella, o use **Ayuda > Licencia...**, para verla.

Edición	Añade
Learning	Cajeras, contornos, taladrado, grabado y acabado 3D en barrido paralelo; todo el gemelo, el editor y la IA
Maker	Desbaste adaptativo, todas las estrategias de acabado 3D, mecanizado de restos, dos caras y varias fases
Pro	Anidado, el optimizador de programas, informes (el informe de inspección, la hoja de amarre), inspección con el palpador comparada con el diseño

Cada edición incluye todo lo de las ediciones que están por encima de ella en la tabla. Una función de una edición superior se muestra con el nombre de su edición. Haga clic en ella para leer qué hace y para empezar una prueba o abrir la licencia.

Cuando termina una prueba o una licencia, Studio vuelve a la edición Learning. Sus proyectos, programas y el código G que generó siguen funcionando.

Empezar una prueba

Puede probar la edición Maker y la edición Pro durante 30 días cada una, una vez por equipo.

1. Haga clic en una función de esa edición, por ejemplo **Optimizar** en el espacio de trabajo **Programa**.
2. Haga clic en **Probar 30 días...**
3. Haga clic en **Empezar la prueba**.

Studio pide la prueba una sola vez al servidor de licencias de Twinwright y envía la edición, un hash de la huella de este equipo y la versión de Studio. Cuando la prueba termina, un aviso en la pestaña **Mensajes** lo indica.

Activar una licencia

Abra **Ayuda > Licencia...**.

Para activar en línea:

1. Escriba el código de activación (`TW-XXXXX-XXXXX-XXXXX-XXXXX`). Mayúsculas, minúsculas, espacios y guiones no importan.
2. Haga clic en **Activar**. Resultado: **Activado.**

Para activar un equipo sin internet:

1. En **Activar sin conexión…**, copie el código de solicitud de este equipo.
2. En un equipo con internet, abra la página de licencias, pegue el código de solicitud y descargue el archivo de licencia (`.twlic`).
3. Lleve el archivo de vuelta y haga clic en **Importar un archivo de licencia…**

Renovar antes de que venza renueva la licencia en línea antes de que termine. Solo envía el número de licencia, un hash de la huella de este equipo y la versión de Studio.

Puestos, pasar una licencia a otro equipo y aulas: Licencias.

Una primera pieza en diez minutos

La primera pieza más rápida de hacer es una de las piezas de ejemplo: Studio prepara por usted el bruto, las herramientas y el origen de pieza.

Abrir la máquina

La máquina ya está abierta: **MC-1 · Mi banco de trabajo**, en el espacio de trabajo **Ejecución**. Las demás máquinas están en **Archivo > Biblioteca de máquinas…**.

Encender y hacer la referencia

Pulse **Referencia** (F3), o la tecla que sugiere la barra de la máquina.

Resultado: los tres ejes buscan su referencia uno tras otro, en unos 20 s. La barra de la máquina muestra **Sin alarma**.

Si la barra de la máquina muestra **Parada de emergencia**, libérela primero (F1) y luego pulse **Encender** (F2) y **Reset** (F12).

Abrir un programa de ejemplo

1. Elija **Archivo > Piezas de ejemplo…**
2. Elija **Placa de motor NEMA 17** (e01_motor_mount): un agujero de ajuste, avellanados cilíndricos y ranuras; la pieza se libera sobre puentes. Resultado: Studio coloca el bruto, carga las herramientas y el programa, y fija el origen de pieza en el bruto. La consola indica cuándo está listo el ejemplo.

Ejecutarlo en el gemelo

1. Pulse **Inicio de ciclo** (F5).
2. Para verlo más rápido, elija **Simulación > Velocidad > ×10 tiempo real**.

La vista 3D muestra la herramienta cortando el bruto. **Retención de avance** (F6) pone el programa en pausa; **Inicio de ciclo** lo reanuda.

Revisar el resultado

- Arrastre en la vista 3D para girarla; la rueda hace zoom.
- **Ver > Coloración > Pieza: desviación respecto al modelo** colorea lo que queda por cortar y cualquier sobrecorte.
- **Análisis** (Alt+5) muestra qué líneas ocuparon el tiempo y las estimaciones del gemelo de la fuerza y de la vibración.

Preparar el bruto

Para su propia pieza, prepare la máquina en **Preparación** (Alt+1) en lugar de cargar un ejemplo:

1. Vaya al paso **Amarre**.
2. En **Pieza y amarre**, elija la **Forma**, el **Material** y el **Utillaje**, y escriba las medidas.
3. Haga clic en **Colocar**.

Vea Preparar un trabajo.

Medir la herramienta

1. Vaya al paso **Herram.**, elija la herramienta y haga clic en **Cambiar herramienta**.
2. Haga clic en **Medir herramienta de ref.**

La máquina debe estar referenciada y el husillo parado.

Fijar el origen de pieza

1. Vaya al paso **Cero** y mueva la herramienta con un desplazamiento manual (jog) hasta el punto que quiera como cero.
2. Haga clic en **Todos a cero** o ponga a cero los ejes uno por uno.

Con un palpador, la tarjeta del palpador encuentra el cero por usted (vea Palpado).

Después abra su programa (**Archivo > Abrir un programa…**) y ejecútelo como antes.

Adónde ir después

- La ventana y los espacios de trabajo: qué hay y dónde.

- Crear programas: el CAM en Studio.
- Ejecutar un programa en el gemelo: qué comprueba y qué estima el gemelo.
- Ejecutar un programa y reanudarlo: ejecuciones, alarmas y volver a empezar.

Un **?** en un panel abre este manual en el capítulo de ese panel. **Ayuda > Documentación** lo abre desde el principio.

La ventana y los espacios de trabajo

La ventana de Studio tiene arriba los menús y una barra superior, en el centro seis espacios de trabajo y abajo la barra de la máquina. Cada espacio de trabajo se ocupa de una tarea principal. La barra de la máquina muestra el estado de la máquina y el siguiente paso dondequiera que se encuentre.

Las partes de la ventana

De arriba abajo:

- la **barra de título**: los menús a la izquierda, el nombre de la ventana en el centro, los botones de la ventana a la derecha;
- la **barra superior**: la máquina, las pestañas de los espacios de trabajo, la edición, el estado de la IA, el nivel de usuario y **?**;
- el **espacio de trabajo**: la vista 3D y los paneles del espacio de trabajo que haya elegido;
- el **panel inferior**, bajo la vista 3D, en los espacios de trabajo que lo tienen;
- la **barra de la máquina**, la misma en todos los espacios de trabajo.

Studio no tiene barra de estado: la barra de la máquina cumple esa función.

La barra de título y los menús

Los menús son **Archivo**, **Edición**, **Ver**, **Máquina**, **Programa**, **Simulación**, **IA**, **Herramientas** y **Ayuda**.

- Pulse Alt y la letra de un menú para abrirlo: Alt+F abre **Archivo**.
- Pulse solo la tecla Alt para resaltar los títulos de los menús y luego use las teclas de flecha, Intro y Esc.

Mientras hay un menú abierto, ninguna tecla llega a la máquina. Un menú muestra solo las opciones de su nivel de usuario (consulte Niveles de usuario).

La barra superior

De izquierda a derecha:

- el logotipo de la marca: abre **Archivo > Biblioteca de máquinas...**;
- la máquina: su modelo, su nombre y su estado. Haga clic en ella, o pulse Ctrl+Mayús+M, para cambiar de máquina: escriba para filtrar y elija con ↑ ↓ y Intro;
- las pestañas de los espacios de trabajo (consulte Los seis espacios de trabajo);
- la edición, por ejemplo **Learning**: haga clic en ella para abrir **Ayuda > Licencia...**;
- el estado de la IA: **IA desactivada**, **IA activada**, **IA conectada** o **La IA espera su aprobación**. Haga clic en él para abrir los ajustes de la IA (consulte El asistente de IA);
- el nivel de usuario: haga clic en él para elegir otro nivel;
- **?**: la lista de atajos de teclado.

La barra de la máquina

La barra de la máquina recorre la parte inferior de la ventana. De izquierda a derecha:

1. **El estado**: **En reposo**, **En ejecución**, **Jog**, **Referenciando**, **Retención**, **Puerta**, **Cambio de herramienta**, **Modo de comprobación**, **Suspendido**, **Alarma**, **Parada de emergencia** o **Apagado**.
2. **El programa**: su nombre, la línea en que está, la sección en que está, una barra de progreso, y el tiempo transcurrido y el tiempo restante. Sin programa muestra **Ningún programa cargado**, si la máquina está referenciada, la herramienta y el decalaje de origen.
3. **El siguiente paso**: **Sin alarma** en verde, o **Siguiente paso:** y la tecla que lo realiza. En una alarma muestra la alarma, el siguiente paso y **Todos los pasos**, que enumera los pasos para volver a trabajar (consulte Alarmas y qué hacer).
4. **El reloj**: el tiempo de la simulación, su velocidad y la velocidad real que alcanza. La tecla del reloj de arena pausa la simulación.
5. **Las teclas**: **Bloque a bloque**, **Inicio de ciclo** (F5), **Retención de avance** (F6) y **Parar** (F7). La tecla que sugiere Studio aparece rellena; una tecla que no puede usar en ese momento aparece atenuada.
6. **Parada** (F1), la tecla roja redonda del extremo derecho.

En una ventana estrecha, la barra omite primero el tiempo del reloj, luego la sección y los tiempos del programa (pasan a la sugerencia), luego las velocidades del reloj y por último las palabras del siguiente paso.

La vista 3D

La vista 3D muestra la máquina, la pieza y la herramienta tal como están ahora.

- Arrastre con el botón izquierdo para girar la vista y con el botón derecho o el central para moverla. La rueda hace zoom hacia el puntero.
- Las teclas de arriba a la izquierda eligen una vista: isométrica, frontal, derecha o superior. **Seguir la herramienta** mantiene la herramienta en el centro. **Mostrar** activa y desactiva el rastro de la punta de la herramienta y la trayectoria del programa.
- **Detalle de herramienta** muestra la herramienta en el corte en una imagen pequeña arriba a la derecha. Aparece en el espacio de trabajo **Ejecución** mientras se ejecuta un programa.
- Las etiquetas de abajo nombran la pieza y la herramienta.
- Haga clic en la pieza o en el rastro: la barra bajo la vista le indica qué línea del programa creó ese punto, y el listado salta a esa línea.

Las mismas opciones están en **Ver > Vista 3D**. **Ver > Coloración** colorea el rastro y la pieza según la velocidad de avance, la fuerza de corte, el espesor de viruta, el índice de vibración o la línea del programa (nivel Ingeniero y superiores).

El panel inferior

El panel inferior está bajo la vista 3D. Sus pestañas dependen del espacio de trabajo:

Espacio de trabajo	Pestañas
Ejecución	**Programa**, **Curvas**, **Consola**, **Mensajes**, **Actividad de IA**
Preparación	**Resultados de la inspección** (en cuanto se ha medido una pieza), **Tabla de herramientas**
CAM	**Resultados de la comprobación**, **Actividad de IA**, **Versiones**, **Vista previa del programa**
Programa	**Consola**

Análisis, **Diagnóstico** y el modo taller no tienen panel inferior.

- **Programa**: el listado del programa. **Seguir la línea** mantiene a la vista la línea que se está ejecutando; las barras de tiempo muestran el tiempo de cada línea en esta ejecución o en la última.
- **Curvas**: las lecturas de la ejecución a lo largo del tiempo.
- **Consola**: lo que dijeron Studio y el controlador. Escriba una línea de código G o un comando `$` en la línea de abajo y pulse Intro para enviarla; ↑ y ↓ recuperan las líneas anteriores.
- **Mensajes**: advertencias y hallazgos, agrupados según su origen. Un número rojo en la pestaña cuenta las advertencias.
- **Tabla de herramientas**: las herramientas de la máquina.
- **Resultados de la inspección**: la última pieza medida con el palpador, comprobación por comprobación: su diseño, la tolerancia, la medición, la desviación y la estimación del gemelo (consulte Inspección).
- **Actividad de IA**: lo que hicieron los clientes de IA en las máquinas de este equipo.

Pulse Ctrl+J, o haga clic en la flecha del extremo derecho de las pestañas, para plegar el panel inferior hasta su fila de pestañas.

Los seis espacios de trabajo

Cambie de espacio de trabajo con las pestañas de la barra superior, con Alt+1 a Alt+6, con Ctrl+Tab y Ctrl+Mayús+Tab, o desde el menú **Ver**. Al pulsar **Inicio de ciclo** en **CAM** o en **Programa**, pasa a **Ejecución**. Mientras se ejecuta un programa, la pestaña **Ejecución** tiene un punto verde.

Preparación

Preparación (Alt+1) deja listos la máquina y la pieza, paso a paso: **Tensión**, **Referencia**, **Amarre**, **Herram.**, **Cero**, **Programa** y **Corte**. El panel de la derecha sigue el siguiente paso hasta que usted elija uno. Consulte Preparar un trabajo; las máquinas, la tensión y la referencia: Máquinas y encendido.

CAM

CAM (Alt+2) crea programas: a la izquierda el amarre, las formas y las operaciones del trabajo; a la derecha los campos del elemento elegido; abajo, **Generar el programa**. Consulte Crear programas. Nivel Ingeniero y superiores.

Programa

Programa (Alt+3) reúne los programas y las piezas de ejemplo, el editor y una vista previa con las comprobaciones del programa. Consulte Programas y el editor.

Ejecución

Ejecución (Alt+4) ejecuta el programa cargado. La columna de la derecha muestra la posición, el husillo y el avance, los overrides, el corte, las teclas del panel de mando, el control manual, y las herramientas y el cero. Consulte Ejecutar un programa y reanudarlo.

Análisis

Análisis (Alt+5) revisa las ejecuciones pasadas: sus curvas, el tiempo y las fuerzas de cada línea, y los lóbulos de estabilidad. Nivel Ingeniero y superiores. Consulte Análisis y diagnóstico.

Diagnóstico

Diagnóstico (Alt+6) muestra la cadena de alimentación de la máquina, las entradas del controlador y los accionamientos, y permite inyectar fallos en el gemelo. Nivel Técnico y superiores. Consulte Análisis y diagnóstico.

Mensajes y qué hacer a continuación

Studio le dice qué ha ocurrido, y qué hacer, en estos lugares:

- la barra de la máquina: el siguiente paso, o la alarma y los pasos para volver a trabajar;
- la pestaña **Mensajes** del panel inferior: advertencias y hallazgos;
- la **Consola**: cada línea del controlador y de Studio;
- **Herramientas > Registro de auditoría…** (Ctrl+L): todo, con quién lo hizo y cuándo.

Máquina > Realizar el paso siguiente realiza el paso que muestra la barra de la máquina, y solo ese paso. Cuando una alarma ha detenido un programa, los pasos terminan con la línea desde la que volver a empezar (consulte Reanudar desde una línea).

Niveles de usuario

Un nivel de usuario muestra lo que una persona necesita en esa etapa y oculta el resto. Hay cinco:

Nivel	Añade
Demo	El espacio de trabajo **Ejecución** con las piezas de ejemplo; las pestañas **Programa** y **Mensajes**
Operador	**Preparación** y **Programa**; los overrides, el panel de mando, el control manual, las herramientas y el cero; los submenús de **Máquina**; la consola, la tabla de herramientas, **Actividad de IA**; los ajustes, el registro de auditoría, el menú **IA**; el cambio de máquina
Técnico	**Diagnóstico**; posicionar la pieza; **Restablecer estado de fábrica...**
Ingeniero	**CAM** y **Análisis**; las lecturas del corte, las curvas, **Coloración**, la exportación de curvas, el modelo con el que comparar
Investigador	La tarjeta **Modelo y datos** en **Diagnóstico**

Cada nivel incluye todo lo de los niveles que están por encima de él en la tabla. Una instalación nueva empieza en **Ingeniero**.

Para cambiar el nivel, use **Herramientas > Nivel de interfaz**, el nivel de la barra superior o **Herramientas > Ajustes... > General**.

> Nota
>
> Un nivel organiza la ventana para enseñar y aprender. No bloquea nada: las teclas de la máquina funcionan en todos los niveles.

Claro, oscuro y modo taller

Ver > Apariencia elige **Como el sistema**, **Claro** u **Oscuro**, y activa o desactiva **Modo taller (grande)**. Las dos elecciones son independientes: el modo taller funciona con cualquiera de los dos aspectos.

Los paneles propios de Studio cambian al instante. Las listas y los cuadros de texto que dibuja el sistema adoptan el nuevo aspecto en el siguiente inicio.

El modo taller está pensado para trabajar junto a la máquina, a un brazo de distancia:

- la barra de la máquina, sus teclas y la parada de emergencia son más grandes, y también el texto y el listado;
- **Ejecución** tiene su propia disposición: la posición, el programa y un listado grande a la izquierda, los indicadores bajo la vista 3D, los overrides, las teclas del panel de mando y los mensajes a la derecha, sin panel inferior;
- la vista 3D tiene dos teclas de vista y no tiene detalle de herramienta.

Los paneles y sus tamaños

Arrastre el borde entre dos paneles para cambiar sus tamaños; haga doble clic en el borde para volver al tamaño de diseño. La vista 3D conserva al menos 360 × 240 píxeles con una escala de pantalla del 100 %. Los bordes de los paneles solo se mueven con el ratón. **Ver > Restablecer la disposición** devuelve de una vez cada panel de cada espacio de trabajo a su tamaño de diseño.

Studio guarda, para cada máquina, los tamaños de los paneles, la pestaña del panel inferior y si está plegado, y el espacio de trabajo en que estaba. Guarda la posición y el tamaño de la ventana para el siguiente inicio.

Máquinas y encendido

Las máquinas de Studio, cómo cambiar de una a otra y cómo se enciende una máquina y se hace su referencia antes de que trabaje.

Máquinas

Cada máquina conserva su propio mundo: su posición, su pieza, sus herramientas y decalajes de origen, sus temperaturas y sus horas de funcionamiento. La máquina de su ventana es la que funciona; las demás esperan tal como se guardaron. En la versión 1.0 solo funciona una máquina a la vez.

Cambiar de máquina

1. Haga clic en la máquina en la barra superior o pulse Ctrl+Mayús+M (**Archivo > Cambiar de máquina…**).
2. Escriba para buscar una máquina, elíjala con ↑ ↓ y pulse Intro. Resultado: Studio guarda la máquina que usted deja y arranca de nuevo con la otra. Esa máquina arranca en frío: enciéndala y haga su referencia.

No puede cambiar de máquina mientras se ejecuta un programa. La lista también muestra las **Máquinas de práctica** que los clientes de IA usan en segundo plano; **Ver (solo lectura)** las muestra en **Actividad de IA**.

La biblioteca de máquinas

Archivo > Biblioteca de máquinas…, o un clic en el logotipo, muestra sus máquinas con su estado, su pieza y su herramienta, y los modelos de máquina. Haga clic de nuevo en el logotipo para volver.

Para la máquina seleccionada:

- **Abrir**: cambiar a ella.
- **Cambiar nombre…**: darle un nombre propio.
- **Copiar como nueva máquina…**: una máquina nueva con el mundo de esta tal como se guardó por última vez.
- **Mostrar su carpeta**: su mundo en el disco.
- **Restablecer estado de fábrica… (dos clics en 5 s)**: la máquina tal como venía (vea Empezar de nuevo con una máquina).

Nueva máquina… añade una máquina del modelo en su estado de fábrica. Las máquinas nuevas, las copias y los restablecimientos de fábrica necesitan el nivel Técnico. La fila **MC-1 real** es un lugar reservado para más adelante: en la versión 1.0 una máquina real se conecta desde la línea de comandos (vea Una máquina real).

Encendido y referencia

Encender la máquina

El gemelo tiene la cadena de alimentación de la MC-1: el interruptor principal QF1 suministra 24 V al controlador, y **Encender** cierra el contactor KM1, que suministra 48 V a los accionamientos y alimenta el variador del husillo. La parada de emergencia abre KM1.

1. Vaya al paso **Tensión**.
2. Si el interruptor principal está apagado, haga clic en **Encender interruptor principal**.
3. Si la tecla indica **Pulsada**, la parada de emergencia está pulsada: haga clic en ella, o pulse F1, para liberarla.
4. Haga clic en **Encender** (F2). Resultado: el panel muestra 48 V conectados y cada accionamiento **Listo**. El siguiente paso es **Referencia**.

Hacer la referencia

Haga clic en **Referencia** (F3). Primero sube Z y después X e Y juntos, cada uno hasta el final de carrera de su extremo del recorrido; en tiempo real esto tarda unos 20 s.

Resultado: **Ejes referenciados: X Y Z** y los límites por software están activados. Hasta que se hace la referencia de la máquina después de encenderla, el controlador está en ALARM:11, y el sensor de longitud de herramienta, el palpador e **Ir al cero** esperan.

Límites por software y recorrido

La MC-1 tiene un recorrido de 200 mm en X, 150 mm en Y y 80 mm en Z. Tras hacer la referencia, la coordenada de máquina de cada eje es 0 en su extremo de referencia: X va de −200 a 0, Y de −150 a 0 y Z de −80 a 0, con el 0 en la parte superior de Z.

- Un desplazamiento manual (jog) que sobrepasaría un límite no se realiza (error:15). Un desplazamiento manual mantenido se detiene justo antes del límite.
- Una línea del programa que sobrepasaría un límite detiene el programa con ALARM:2. La posición se conserva: **Reset**, **Desbloquear** y luego compruebe el programa y el origen de pieza.
- Un eje que alcanza un final de carrera provoca ALARM:1: **Reset**, **Desbloquear**, salga del final de carrera con un desplazamiento manual y luego **Referencia**.

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Preparar un trabajo

El espacio de trabajo **Preparación** deja listos la máquina y la pieza, paso a paso: la tensión, la referencia, el bruto y cómo se sujeta, las herramientas y el origen de pieza. Cada paso se muestra primero en el gemelo, de modo que usted ve el amarre antes de que se mueva nada. Las máquinas, la tensión y la referencia tienen un capítulo propio: Máquinas y encendido.

El espacio de trabajo Preparación

Preparación (Alt+1) tiene los pasos a la izquierda, la vista 3D en el centro y el panel del paso elegido a la derecha. El panel inferior, bajo la vista, contiene la **Tabla de herramientas**.

Los pasos

Paso	Su panel	Está hecho cuando
Tensión	El interruptor principal, **Encender**, **Reset**, **Desbloquear**	El controlador funciona y los accionamientos están alimentados
Referencia	**Referencia**, **Desbloquear**, **Reset**	Los tres ejes están referenciados
Amarre	El bruto, cómo se sujeta y dónde	Hay un bruto montado, o la mesa se deja vacía
Herram.	La herramienta en el husillo, el sensor de longitud de herramienta	Se ha medido la herramienta de referencia, se ha cargado una tabla de herramientas o se ha fijado el origen de pieza en Z
Cero	El desplazamiento manual (jog), el origen de pieza, la tarjeta del palpador	Se ha fijado un decalaje de origen
Programa	**Abrir un programa…**, **Piezas de ejemplo…**, **Comprobar el programa**	Hay un programa cargado
Corte	**Inicio de ciclo**, **Bloque a bloque**	El programa se ejecutó hasta el final

Un paso hecho tiene una marca verde; el siguiente paso tiene una pequeña flecha al final. El panel muestra el siguiente paso hasta que usted hace clic en otro; haga clic en el siguiente paso para que el panel vuelva a seguirlo. La línea bajo los pasos dice qué hacer ahora. El **?** de un panel abre su parte de este manual.

La vista 3D durante la preparación

Mientras cambia los campos del bruto, el bruto que está montando se dibuja transparente sobre la mesa: en verde cuando se puede montar así, en rojo cuando no. **Colocar** lo pone ahí; **Revertir** o Esc descarta el cambio.

El bruto

Forma, material y dimensiones

En el paso **Amarre**, en la tarjeta **Pieza y amarre**:

1. Elija la **Forma**: **Bloque**, **Barra redonda (vertical)**, **Tubo (vertical)**, **Barra hexagonal (vertical)**, **Barra redonda (horizontal)** o **Archivo STL (importar la forma)…**.
2. Elija el **Material**, o escriba su nombre de otra norma (C45, EN AW-6061, SUS304): la lista muestra la calidad cuyos datos de corte toma Studio, y el amarre conserva el nombre que usted escribió.
3. Escriba las dimensiones que necesita la forma: **Longitud L**, **Ancho W**, **Altura H**, **Diámetro D**, etc.
4. Elija cómo se sujeta y dónde (más abajo) y luego haga clic en **Colocar**. Resultado: el panel indica qué hay en la mesa y su masa.

Las dimensiones están en la unidad de visualización; en pulgadas, los campos también admiten fracciones (consulte Fracciones de pulgada).

> **Nota**
>
> Un nombre de otra norma es una calidad próxima: los datos de corte proceden de la calidad más cercana de la lista. El material es el que indica el certificado del proveedor.

Un bruto a partir de un archivo STL

Elija **Archivo STL (importar la forma)…** en la lista de formas y seleccione el archivo (en mm). Studio apoya la forma en su punto más bajo y la centra; su caja envolvente da las dimensiones. El material bajo una zona en voladizo cuenta como sólido. Si no se puede leer el archivo, la consola indica por qué.

Sujeción del bruto

Elija el **Utillaje**: **Bridas · ambos extremos**, **Bridas · ambos lados**, **Bridas · cuatro lados**, **Mordaza**, **Bloque en V (barra en horizontal)**, **Cinta de doble cara**, **Sin fijar (solo apoyada)** o **Fijación ideal (para scripts)**.

El bruto y su utillaje deben quedar sobre la mesa (300 × 220 mm) y no interferir con el sensor de longitud de herramienta, situado detrás a la izquierda. Mientras se ejecuta un programa, el gemelo compara la fuerza de corte con lo que sujeta el utillaje: avisa cuando el margen es pequeño, y un bruto que se desliza se mueve en el gemelo, de modo que lo que se corta después queda desviado.

Mordaza y paralelas

Sobre las bocas es lo que sobresale la cara superior del bruto por encima de las bocas: 5 mm si no lo cambia. Studio elige las paralelas que dan como mínimo esa altura. La mordaza se abre de 3.25 a 77 mm, y las bocas deben sujetar al menos 2 mm del bruto.

Bridas

Bridas de sujeción sobre espárragos en los agujeros roscados de la mesa (M6, en una cuadrícula de 25 mm): **ambos extremos** a lo largo del bruto, **ambos lados** a lo ancho, **cuatro lados** en las dos direcciones. Cada brida necesita un agujero a 7–35 mm fuera del borde del bruto; si no hay ninguno, el panel nombra la brida que no encuentra agujero. Para una barra horizontal use el bloque en V o la mordaza.

Cinta de doble cara

La cinta sujeta el bruto por su superficie sobre la mesa. Úsela para cortes ligeros y piezas planas, no para una barra horizontal. Una vez que cede, solo la fricción sujeta el bruto. **Sin fijar (solo apoyada)** sujeta solo por el peso y la fricción; **Fijación ideal (para scripts)** nunca se desliza, para scripts y pruebas.

Espacio sobre el bruto

Los programas que escribe el CAM usan el espacio sobre el bruto de esta manera:

- Un programa empieza, termina y cambia de herramienta en la parte alta del recorrido de Z.
- Entre operaciones, la herramienta se desplaza a la altura de seguridad: 5 mm sobre la cara superior del bruto, o 2 mm sobre una brida que sobresale más que el bruto.
- Donde el amarre deja menos espacio que eso (una herramienta larga, una mordaza alta, un bruto alto), Studio baja la altura de seguridad hasta lo que haya, como mínimo 0.3 mm por debajo del extremo superior del recorrido, y lo indica junto con el programa. Cuando eso dejaría menos de 0.5 mm sobre el bruto, no escribe el programa y explica por qué.

La tabla **Espacio libre en Z** de la hoja de amarre comprueba el palpador y cada herramienta para cada fase. Consulte Espacio libre en Z.

Colocación del bruto

Posición X e **Y** sitúan el centro del bruto en las coordenadas de mesa del gemelo (el centro de la mesa es X 100, Y 75). **Ángulo °** lo gira en sentido antihorario, visto desde arriba.

- **Colocar**: un bruto nuevo ahí. El bruto anterior y lo que se cortó en él desaparecen.
- **Mover aquí**: el mismo bruto movido, girado o sujeto de otra manera. Lo que se cortó se conserva.
- **Retirar**: el bruto fuera de la mesa.
- **Revertir**: los campos vuelven a lo que hay en la mesa.

El montaje espera mientras el husillo gira, la máquina se mueve o se ejecuta un programa.

Posicionamiento en la vista 3D

Haga clic en **Posicionar pieza** (nivel Técnico y superiores) o use **Ver > Vista 3D > Posicionar pieza**. Después:

- arrastre el bruto para moverlo, el tirador naranja para girarlo y los tiradores azules para cambiar su tamaño (con Mayús en pasos finos, con Ctrl en pasos gruesos);
- o use el teclado: las teclas de flecha lo mueven 1 mm (Mayús 0.1 mm, Ctrl 10 mm), Q y E lo giran 15° (Mayús 1°), Intro lo monta y Esc lo devuelve a su sitio.

Salir del espacio de trabajo **Preparación** termina el posicionamiento.

Voltear el bruto

Voltear da la vuelta al bruto en su utillaje, 180° alrededor de su longitud: la parte superior y la inferior se intercambian, y también la delantera y la trasera. Lo que se cortó gira con él. Los decalajes de origen se quedan como estaban, así que fije de nuevo el origen de pieza (consulte Palpado).

Para una pieza mecanizada por dos caras, el CAM planifica el volteo entre sus fases (consulte Trabajos con varias fases).

Herramientas y tabla de herramientas

La **Tabla de herramientas** del panel inferior enumera las herramientas de la máquina: el número T, la herramienta y la especificación que lee el programa. La herramienta que está en el husillo se marca con **En el husillo**. La tabla procede de una pieza de ejemplo, de un trabajo de CAM o de un cliente de IA; no se puede cambiar en la ventana.

Con una tabla de herramientas, el programa cambia de herramienta por sí solo (`M6`): la herramienta entra, la máquina la mide en el sensor de longitud de herramienta y continúa. Sin tabla, un `M6` espera a que usted ponga la herramienta (**Cambiar herramienta** en el paso **Herram.**) y luego **Inicio de ciclo**.

Cambiar la herramienta del husillo

1. Vaya al paso **Herram.**
2. Elija la herramienta en la lista **Herramienta**, por ejemplo **Fresa plana Ø6, 2 filos (voladizo 15.5)**.
3. Haga clic en **Cambiar herramienta**.

El husillo debe estar parado y la máquina en reposo. Si la herramienta nueva es más larga o más corta, mídala (**Fijar compensación**) o fije de nuevo el origen de pieza en Z. La lista contiene las herramientas estándar, primero las métricas (primero las de pulgada cuando Studio muestra pulgadas). Cómo se describe una herramienta, por ejemplo `flat 6 n2 L15.5`: Herramientas y tabla de herramientas.

Medir herramientas con el sensor de longitud de herramienta

El sensor de longitud de herramienta está detrás a la izquierda de la mesa. La máquina debe estar referenciada y en reposo, y el husillo parado.

1. Con la herramienta con la que fijó el origen de pieza en Z puesta en el husillo, haga clic en **Medir herramienta de ref.**
2. Después de cada cambio de herramienta, haga clic en **Fijar compensación**. Resultado: Studio mide la herramienta nueva y fija su compensación de longitud (`G43.1`) a partir de la diferencia.

Las mismas teclas están en **Máquina > Sensor de longitud de herramienta y palpador**. La referencia se conserva hasta que se cierra Studio: mídala de nuevo después de reiniciar.

El modelo con el que comparar

Con el diseño de la pieza, el gemelo muestra dónde se aparta de él la pieza cortada (nivel Ingeniero y superiores).

1. Fije primero el origen de pieza: el modelo se coloca en el cero del programa cuando lo carga.
2. En el paso **Amarre**, bajo **Comparación con el modelo**, haga clic en **STL del modelo (comparar)…** y elija el archivo STL de la pieza (en mm, con su origen en el cero del programa). Resultado: la pieza se colorea según su desviación respecto del modelo (**Ver > Coloración > Pieza: desviación respecto al modelo**): dentro de ±20 µm se considera bueno; los demás colores muestran material sobrante y sobrecortes.

Dejar de comparar quita el modelo. Las piezas de ejemplo cargan el suyo.

Guardar y abrir amarres

- **Guardar amarre…** escribe el amarre de la tarjeta (la forma, las dimensiones, el material, el utillaje y la posición) en un archivo JSON.
- **Abrir amarre…** carga uno de nuevo en la tarjeta; haga clic en **Colocar** para montarlo. También abre un archivo STL como forma del bruto.
- **Exportar pieza…** guarda la pieza tal como está cortada, como STL (en coordenadas de programa, de bruto o de máquina) o como mapa de alturas de 16 bits (PNG). Lo mismo está en el menú Archivo: **Archivo > Exportar pieza…**, desde cualquier espacio de trabajo.

Cada máquina conserva su amarre y su pieza cortada en su mundo. Studio lo guarda cada vez que la máquina lleva dos segundos parada después de un cambio, y al cerrar la ventana; así, tras un bloqueo o un corte de luz, la máquina vuelve tal como estaba en su última parada.

Trabajos con varias fases

Una pieza que se mecaniza por dos caras, o que se sujeta de dos maneras, es un trabajo con varias fases. En el espacio de trabajo **CAM**, **Fase…** añade una fase al trabajo. El árbol del trabajo tiene entonces un nivel por cada fase, con sus operaciones y su programa debajo, y un paso **Volteo** entre dos fases. Un trabajo con una sola fase no tiene ese nivel.

- Cada fase tiene sus propias operaciones y su propio programa. Los programas se nombran con el nombre del trabajo y el número de la fase, por ejemplo `plate-1.nc` y `plate-2.nc`.
- Una fase parte de la pieza tal como la dejaron las fases anteriores. En el gemelo, Studio ejecuta la fase 1, voltea la pieza como indica el paso **Volteo** y escribe el programa de la fase siguiente para esa pieza. En la vista 3D, lo que se cortó antes y ahora queda debajo se dibuja con líneas discontinuas.
- El árbol llama a una fase frontal o trasera según las veces que se ha volteado la pieza.
- Bajo el panel de una operación, **Esta fase** muestra el utillaje de la fase, su cero, cómo se volteó la pieza y el programa.
- **Resultados de la comprobación** tiene una fila por cada fase y otra para el volteo. **Total** suma el tiempo en la máquina y da el peor índice de vibración y el mayor sobrecorte. También compara la pieza terminada, tras la última fase, con el modelo.

Un trabajo tiene como máximo 8 fases. Un trabajo anidado todavía no puede tener fases. Varias fases están disponibles en la edición Maker y superiores (consulte Ediciones).

La hoja de amarre tiene una sección para cada fase, con lo que hay que hacer entre ellas. Consulte Hojas de amarre.

Voltear la pieza entre fases

Seleccione **Volteo** en el árbol del trabajo. Su panel dice cómo llega la pieza a la fase siguiente. Los ajustes pertenecen a la fase que sigue al paso.

Ajuste	Opciones
Tipo de volteo	**Alrededor de U, 180°**: los extremos izquierdo y derecho se quedan donde están; delante y detrás se intercambian, y también arriba y abajo. **Alrededor de V, 180°**: los bordes delantero y trasero se quedan; izquierda y derecha se intercambian, y también arriba y abajo. **Sin voltear (cambio de sujeción)**: la pieza se queda como está y solo cambia el utillaje.
Origen	**Esquina superior delantera izquierda** o **Centro de la cara superior**: dónde se vuelve a encontrar el origen de pieza (consulte Trasladar el origen de pieza)
Sujeción	Cómo se sujeta la pieza a partir de esta fase: el mismo utillaje u otro

U va a lo largo del eje X de la máquina y V a lo largo de su eje Y, a través de la pieza tal como está en la fase anterior.

Qué hace el operador enumera los pasos en la máquina: dejar que termine el programa de la fase anterior (husillo parado, refrigerante desactivado), soltar la pieza, voltearla, sujetarla, fijar de nuevo el origen de pieza y cargar el programa siguiente. **En el gemelo** dice qué hace el gemelo con la pieza.

> **Precaución**
>
> Voltee la pieza real como indica el paso. El programa siguiente se comprueba en el gemelo solo para la pieza volteada de esa manera. Una pieza volteada alrededor del otro eje, o sin voltear, se corta donde no se comprobó nada.

Trasladar el origen de pieza

Cuando se voltea la pieza o se sujeta con otro utillaje, sus esquinas ya no están donde estaban: hay que volver a encontrar el origen de pieza (G54). El árbol del trabajo lo indica junto a la fase, y el programa de la fase parte del cero que usted eligió:

- **Esquina superior delantera izquierda**: la esquina delantera izquierda de la cara superior de la pieza, tal como queda ahora. Este es el cero después de un volteo o de un cambio de utillaje, salvo que usted elija otro.
- **Centro de la cara superior**: el centro de la cara superior de la pieza, por ejemplo para una pieza redonda.

Encuentre el cero con el palpador (consulte Palpado). La hoja de amarre da los pasos de palpado de cada fase.

Cuando la pieza no se voltea ni se sujeta de otra manera, se conserva el cero de la fase anterior y no se vuelve a encontrar. Una segunda fase que solo usa otras herramientas funciona así; su sección de la hoja de amarre dice que el origen de pieza no se vuelve a tocar.

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Palpar y buscar el origen de pieza

Con un palpador en el husillo, Studio encuentra un borde, una esquina, el centro de un agujero o de un saliente, o la cara superior, y fija a partir de ello el origen de pieza. La tarjeta del palpador comprueba el ciclo con el amarre antes de que empiece, y el gemelo lo ejecuta como lo haría la máquina.

La tarjeta del palpador está en **Preparación**, paso **Cero**, bajo las teclas de desplazamiento manual (jog). En la versión 1.0, la tarjeta del palpador funciona en el gemelo. En una máquina real, palpe con el propio control de la máquina.

Antes de empezar

Necesita:

- el bruto montado en la mesa (paso **Amarre**);
- el palpador en el husillo (consulte El palpador en la tabla de herramientas);
- el husillo parado.

La máquina no tiene que estar referenciada: un ciclo que encuentra la máquina en alarma o sin referenciar la reinicia y hace primero la referencia, y la tarjeta lo indica.

> **Precaución**
>
> En la MC-1, la mordaza con el palpador de 30 mm no deja espacio sobre el bruto (consulte Espacio sobre el bruto). Sujete el bruto con cinta de doble cara o con bridas cuando palpe.

El palpador en la tabla de herramientas

1. En **Preparación**, vaya al paso **Herram.**
2. En **Herramienta**, elija **Palpador 3D, bola Ø2 (voladizo 30, para localizar bordes)**.
3. Haga clic en **Cambiar herramienta**.

Resultado: la primera línea de la tarjeta del palpador, **En el husillo**, nombra el palpador. Cuando el husillo tiene otra herramienta, la tarjeta dice que primero hay que poner el palpador.

En la descripción de una herramienta, el palpador se escribe `probe 2 L30`: una bola de 2 mm, con 30 mm que sobresalen del portaherramientas.

Espacio sobre el bruto: el presupuesto de Z

Con Z en la parte alta de su recorrido, la punta del palpador tiene que pasar por encima del punto más alto del amarre: la cara superior del bruto, las bocas de la mordaza o las bridas. Studio lo calcula para la herramienta que está en el husillo y para cada herramienta de la tabla de herramientas cada vez que cambian el amarre o las herramientas. Cuando una herramienta no consigue pasar sobre el amarre, el paso **Amarre** dice cuánto le falta y sugiere una herramienta más corta o un amarre más bajo (bridas en lugar de la mordaza).

La tarjeta del palpador comprueba lo mismo para el ciclo: mueve el palpador a una altura segura sobre el bruto y el utillaje, y muestra esa altura en las comprobaciones.

La tarjeta del palpador

De arriba abajo:

- **En el husillo**: el palpador y si está calibrado (**Calibrado** con la hora, o **Sin calibrar**);
- **Ciclos de palpador**: los seis ciclos y los campos del elegido;
- las comprobaciones;
- **Ver el recorrido en 3D** e **Iniciar palpado…**.

Después de un ciclo, la tarjeta muestra su resultado en su lugar (consulte Fijar el origen de pieza con lo encontrado). Bajo la tarjeta del palpador está la tarjeta **Medir la pieza** y, debajo, la tarjeta **Informes** (consulte Inspección).

Los campos numéricos aceptan la unidad de visualización y, en pulgadas, también fracciones. Un campo que no se puede leer se pone rojo e indica por qué (consulte Escribir números).

Las comprobaciones

La tarjeta comprueba el ciclo mientras usted lo prepara, antes de que se mueva nada. Cada comprobación es verde (✓), ámbar (!) o roja (✕). Cualquier comprobación roja mantiene en gris **Iniciar palpado…**.

Comprobación	Verde	Ámbar o roja
El palpador	**Palpador en el husillo, sin disparar**	No hay palpador en el husillo; la entrada del palpador ya está disparada
La máquina	**Husillo parado, máquina referenciada y en reposo**	Ámbar: en alarma o sin referenciar, el ciclo hace primero la referencia. Roja: el husillo gira o la máquina está ocupada
Los límites de software	**Cada movimiento dentro de los límites de software**	Un movimiento queda fuera de ellos

Comprobación	Verde	Ámbar o roja
La altura de seguridad	**Altura de seguridad Z …**, y a qué distancia sobre el bruto y el utillaje	No hay espacio sobre el bruto para pasar por encima
La profundidad	**Profundidad … bajo la cara superior**, y, para un bruto en la mordaza, cuánto sobresale por encima de las bocas	La bola bajaría por debajo de las bocas, o demasiado cerca de la cara inferior del bruto

Cuando el gemelo sigue a una máquina real, una línea roja dice que se palpe allí con el propio control de la máquina.

Probar primero un ciclo en el gemelo

Las comprobaciones salen de una prueba en seco: el gemelo calcula el ciclo sin mover nada. **Ver el recorrido en 3D** dibuja sus movimientos en la vista 3D, en lugar de la trayectoria del programa, hasta que usted sale del paso.

Cada toque se hace de la misma manera:

1. un movimiento protegido hasta el inicio (`G38.3`, F600);
2. un toque rápido (`G38.2`, F300);
3. una retirada;
4. un toque lento (F20): el resultado es el punto donde toca este.

Ciclos de palpado

Elija un ciclo, rellene sus campos y haga clic en **Iniciar palpado…**. Cada ciclo empieza a 3 mm de donde espera la cara y busca 5 mm más allá. **Profundidad** es cuánto por debajo de la cara superior del bruto toca el centro de la bola (3 mm si no lo cambia). **Inicio**: **Del bruto** parte del bruto del amarre; **X Y dados** parte de un punto que usted da en coordenadas de pieza.

Borde

Encuentra una cara del bruto. **Qué cara**: **Izquierda (X−)**, **Derecha (X+)**, **Delante (Y−)** o **Detrás (Y+)**. **Puntos**: **1 punto** o **2 puntos (también el giro)**, que además da cuánto está girado el bruto.

Resultado: **Cara X** o **Cara Y** y, con dos puntos, **Giro**.

Esquina

Encuentra una esquina a partir de dos caras. **Qué esquina**: **Delante izquierda**, **Delante derecha**, **Detrás izquierda** o **Detrás derecha**; **Exterior** es la esquina del bruto, **Interior** la esquina de una cajera (indique su posición con **X Y dados**). **Desde la esquina** es la distancia a la esquina a la que se toca cada cara (5 mm). **3 puntos (también el giro)** toca dos veces la cara X y da también el giro.

Resultado: **Esquina X**, **Esquina Y** y, con tres puntos, **Giro**.

Agujero y saliente

Centro del agujero palpa un agujero desde su centro hacia fuera; **Centro del saliente** palpa un saliente desde fuera, pasando por encima de su cara superior. **Puntos**: **4 puntos** u **8 puntos**. **Diámetro** es la medida nominal; si se deja vacío, se toma de un bruto redondo. Un bruto en forma de bloque necesita el diámetro y **X Y dados**.

Resultado: **Centro X**, **Centro Y** y **Diámetro**.

Cara superior

Superficie Z toca la cara superior una vez, en el centro del bruto o en **X Y dados**.

Resultado: **Superficie Z**.

Fijar el origen de pieza con lo encontrado

La tarjeta de resultados muestra cada valor en coordenadas de máquina y de pieza, cada toque y cómo se aplicó la calibración del palpador. Cuando el bruto está girado, indica cuánto; puede fijar el ángulo del amarre a ese valor en el paso **Amarre**.

Para convertir el resultado en el origen de pieza:

1. Haga clic en **Fijar el cero…**
2. Marque los ejes que se van a fijar. Los ejes que el ciclo no midió no se pueden marcar: una esquina no da Z. Resultado: el cuadro de diálogo muestra la línea que enviará, por ejemplo `G10 L2 P1 X-129.982 Y95.011`.
3. Haga clic en **Fijar como cero G54** (o en el decalaje de origen que estaba en uso).

El cero queda en lo que se encontró: la cara, la esquina, el centro o la superficie. **Copiar el programa** copia las líneas que envió el ciclo. **Medir de nuevo** vuelve a los campos.

> **Nota**
>
> Los números del gemelo muestran el método y el flujo. No son la precisión de medición de esta máquina.

Calibrar el palpador con un anillo patrón

Un palpador se dispara un poco después de tocar (precurso), y no igual en todas las direcciones. Calibrar con un anillo patrón mide esto, y los ciclos posteriores se corrigen con ello.

1. Coloque el anillo patrón en la mesa y elija **Calibración con anillo**.
2. Escriba el **Diámetro del anillo** como indica el certificado del anillo.
3. **Posición**: **Del bruto (el agujero del tubo)** cuando el anillo está montado como bruto, o **Dado**, con su centro.
4. Haga clic en **Iniciar calibración…** Resultado: **Radio efectivo de la bola**, **Precurso XY**, **Lobulado triple** y **Centro del anillo**.
5. Haga clic en **Guardar para esta máquina** o en **No guardar**.

La calibración pertenece a la máquina y a ese diámetro de bola, y va con el archivo de mundo de la máquina. Un restablecimiento de fábrica la olvida. Cada resultado indica qué calibración usó. Mientras no calibre, la tarjeta muestra **Sin calibrar**: un agujero se mide entonces con dos precursos de más, y un saliente con dos de menos.

La máquina guarda una sola calibración. Si pone un palpador con otra bola (un vástago Ø0.5 para agujeros pequeños, por ejemplo), la tarjeta muestra **Sin calibrar** e indica la bola y el anillo con que se hizo la calibración de la máquina: calibre la nueva bola en el anillo patrón. Esa calibración sustituye a la anterior; al volver a la primera bola, calíbrela de nuevo.

Cuando un ciclo se detiene

Cuando un ciclo no puede terminar, la tarjeta de resultados dice por qué en rojo, la consola lo registra y **Medir de nuevo** le lleva de vuelta a los campos. Los motivos habituales:

- un toque no encontró nada dentro de su búsqueda: la cara no está donde la esperaba el ciclo. El controlador genera ALARM:5;
- el palpador tocó algo de camino a un toque: hay algo donde el bruto no debería estar;
- los toques no forman un círculo (agujero, saliente, anillo).

Tras una alarma, siga los pasos de la barra de la máquina: **Reiniciar**, luego **Desbloquear** (consulte Alarmas y qué hacer). El palpador puede estar en cualquier sitio cuando el ciclo se detuvo: mire la vista 3D antes de hacer un desplazamiento manual.

> **Precaución**
>
> Si el palpador toca mientras la máquina está en retención de avance, pulse **Reset** y vuelva a palpar. **Inicio de ciclo** seguiría avanzando más allá del contacto.

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Crear programas (CAM)

Studio calcula las trayectorias de herramienta por sí mismo. Usted describe la pieza como elementos: una cajera, un contorno, agujeros, un grabado. Studio elige los datos de corte con los modelos físicos del gemelo y escribe el código G. Después ejecuta el programa en el gemelo antes de que usted corte.

Qué hace el CAM

- Las formas son datos. La forma de un elemento es un rectángulo, un círculo, un polígono, un trazado de un archivo SVG o DXF, texto, o una malla (STL, 3MF, OBJ) para trabajo en 3D.
- Las trayectorias de herramienta se calculan. Studio desfasa la forma, vacía el área y enlaza los movimientos.
- El gemelo elige los datos de corte. La velocidad, el avance y la profundidad salen del material y luego de los modelos del gemelo para la potencia del husillo, las fuerzas de corte, la vibración y la flexión de la herramienta. Son estimaciones.
- El gemelo comprueba el programa. Studio puede ejecutar el programa en el gemelo antes de que usted corte, y cambiar lo que la ejecución muestra que es excesivo.

Todo se ejecuta en su equipo. Crear un programa no necesita conexión a Internet.

La forma más rápida es describir la pieza al asistente de IA, por ejemplo:

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

El asistente usa la herramienta `program_generate`. Indica qué hizo, cuánto tarda y qué encontró el gemelo.

Un trabajo: elementos, herramientas y material

Un trabajo es lo que Studio guarda sobre la pieza:

- su nombre, que es también el nombre del programa, por ejemplo `bracket.nc`;
- el material, por ejemplo 6061-T6 (se entienden los nombres de otras normas, como EN AW-6061);
- la herramienta en el husillo y la tabla de herramientas;
- los elementos, en el orden en que se cortan.

Cada programa que escribe Studio se guarda como una versión, con el trabajo que lo creó, en la carpeta de programas. No se pierde nada cuando lo vuelve a crear: las versiones anteriores permanecen.

El asistente puede abrir un trabajo y modificarlo, también el que usted creó en el espacio de trabajo CAM, por ejemplo:

```
Muéstreme el trabajo de bracket.nc; luego haga su cajera de 3 mm de profundidad y quite el segundo taladrado.
```

Usa la herramienta `job_get` para leer el trabajo, con una línea por cada operación, y `job_edit` para cambiar valores y quitar, añadir o mover operaciones. El trabajo modificado se vuelve a crear como la siguiente versión del programa y se comprueba en el gemelo como antes. Cuando un cambio no se puede hacer, no se cambia nada y el asistente indica por qué. Tras los intentos propios del gemelo (el bucle cerrado que elimina la vibración), el trabajo que lee el asistente es el de la última versión, con lo que cambiaron esos intentos.

Operaciones

Operación	Qué corta	Herramienta
Planeado	La cara superior del bruto, plana	Fresa
Cajera	Un área hasta una profundidad: rectángulo, círculo, polígono o una forma	Fresa
Contorno	A lo largo de un perfil: por fuera, por dentro o sobre él	Fresa
Ranura	Una ranura de un ancho dado a lo largo de una línea	Fresa
Taladrado	Agujeros en puntos, con picado cuando son profundos	Broca
Mandrinado	Agujeros mayores que la herramienta, siguiendo una hélice	Fresa
Escariado	Agujeros taladrados, llevados a medida	Escariador
Roscado	Roscas interiores	Fresa de roscar
Ranura interior	Una ranura anular en la pared de un agujero: un anillo elástico, una junta tórica, una trampa de luz	Fresa para ranuras en T
Chaflán	El borde de un perfil	Fresa en V o fresa de chaflanar
Grabado	Texto o líneas	Fresa en V o fresa pequeña
Tallado en V	Texto o formas, con la profundidad según el ancho	Fresa en V
Perlage	Círculos que se solapan sobre una cara, la decoración de los relojeros	Fresa plana tan ancha como un círculo
Desbaste 3D, acabado 3D	Una malla	Fresa o fresa esférica

En el espacio de trabajo CAM, el panel tiene formularios para planeado, cajeras, contornos y chaflanes (una forma importada, o un rectángulo, círculo o polígono indicado en el panel), ranuras, taladrado, mandrinado, fresado de roscas y grabado. Un trabajo con otras operaciones —de un cliente de IA o de un ejemplo— se abre igualmente: esas operaciones se conservan tal como las da el trabajo, y se generan y comprueban con las demás. Al seleccionarlas, el panel lista sus ajustes

con los nombres de program_generate (longitudes en mm); se cambian con un cliente de IA o en el archivo del trabajo.

Planeado, cajeras y contornos

Una cajera se vacía por capas. Su estrategia fija la trayectoria en cada capa:

- anillos (el valor por defecto): anillos desde el centro hacia la pared;
- zigzag: pasadas de ida y vuelta;
- adaptativa: una trayectoria que mantiene uniforme el contacto de la herramienta, para cajeras profundas en materiales más duros.

Las islas de una cajera se dejan en pie. El mecanizado de restos vacía solo lo que dejó una herramienta mayor anterior.

Un contorno corta a lo largo de un perfil, una capa cada vez, en fresado en concordancia. Los puentes de sujeción pueden sujetar una pieza que el contorno recorta.

Ranuras

Una ranura se corta a ancho completo por capas y luego se amplía hasta su ancho. Una ranura más ancha que la herramienta puede cortarse de forma trocoidal: pequeños bucles a lo largo de su línea, con menos carga sobre la herramienta.

Agujeros: taladrado, mandrinado, escariado y roscas

- El taladrado usa el ciclo de taladrado del controlador. Un agujero más profundo que tres veces el diámetro de la broca se hace con picado.
- El mandrinado corta un agujero mayor que la fresa siguiendo una hélice. Hasta el doble del diámetro de la herramienta.
- El escariado lleva a medida un agujero taladrado con un escariador del ajuste del agujero. Taladre primero el agujero.
- El fresado de roscas corta una rosca interior con una fresa de roscar, de abajo hacia arriba. Las roscas métricas (M8), unificadas (1/4-20 UNC), de tubería (G1/4) y cónicas de tubería (1/4 NPT, Rc1/4) se reconocen por su nombre.

Chaflanes, grabado y tallado en V

Un chaflán sigue el borde de un perfil con una fresa en V, con el ancho que usted indique. El grabado sigue líneas o los trazos de un texto a una sola profundidad; el texto empieza en su X, se centra en ella o termina allí (**Alineación** en el panel). El tallado en V corta una forma con una fresa en V: la profundidad sigue el ancho de la forma, de modo que las esquinas salen vivas.

Desbaste y acabado 3D

Una malla (STL, 3MF u OBJ) se desbasta por niveles y luego se acaba. Las estrategias de acabado son paralela, Z constante, pendiente y llano y lápiz.

Herramientas y tabla de herramientas

Una herramienta se escribe con su tipo, su diámetro y lo demás que necesita, por ejemplo:

Especificación	Herramienta
`flat 6 n3 L20`	Fresa plana, 6 mm, 3 filos, 20 mm de voladizo
`ball 3 L12`	Fresa esférica, 3 mm
`drill 3.4 118 L20`	Broca, 3.4 mm, punta de 118°
`v 6 90 L18`	Fresa en V, 6 mm, 90°
`reamer 6 H7 n6 L32`	Escariador de máquina, 6 mm, para agujeros H7
`thread 5 60`	Fresa de roscar, 5 mm, perfil de 60°
`flat 2 n2 s4 l6 L10`	Fresa plana, 2 mm, con mango de 4 mm que empieza 6 mm por encima de la punta
`thread 2 60 d1.4 n1 s4 l6 L9`	Fresa de roscar, cabeza de 2 mm, cuello de 1.4 mm hasta 6 mm, mango de 4 mm por encima

Indique el voladizo (`L`) de cada herramienta. Studio comprueba entonces que el portaherramientas no llegue al bruto, y el Espacio libre en Z de la hoja de amarre puede evaluar la herramienta.

Para una fresa pequeña con mango escalonado y para una fresa para entallas (de roscar, para ranuras en T, de cola de milano o de bola completa), indique también el diámetro del mango (`s`) y a qué altura sobre la punta empieza el mango (`l`). Donde el mango rozaría las paredes del corte, Studio no corta más hondo que `l`. El gemelo comprueba si el mango y el cuerpo del palpador chocan con la pieza o con el utillaje.

La tabla de herramientas enumera las herramientas por sus números. Un elemento nombra la herramienta que usa, y Studio añade los cambios de herramienta (`T2 M6`).

Datos de corte del gemelo

Studio parte de los datos de corte del material para la herramienta. Luego usa los modelos del gemelo:

- la potencia del husillo: el corte usa como máximo alrededor del 80 % de ella;
- la vibración: la profundidad y la velocidad se mantienen donde el modelo de estabilidad del gemelo espera un corte estable;
- el utillaje: la fuerza del corte se mantiene muy por debajo de lo que sujeta el utillaje;
- la flexión de la herramienta: las pasadas de acabado se mantienen ligeras.

Con la verificación activada, Studio ejecuta el programa en el gemelo. Si la ejecución muestra vibración o una carga demasiado alta, Studio cambia el programa, por ejemplo con capas menos profundas, y lo intenta de nuevo. Cada versión que prueba se conserva. La respuesta indica qué se cambió y por qué.

Son estimaciones de modelos físicos. Observe y escuche el primer corte en la máquina real.

Ajustes y tolerancias

Escriba una medida con su ajuste, por ejemplo `"d": "Ø12 H7"` para un agujero o `"fit": "g6"` para un eje. Studio corta la medida hasta el centro de la zona del ajuste y acaba la pared en una pasada propia.

Los ajustes habituales pueden nombrarse: `sliding` (H7/g6), `locating` (H7/h6), `transition` (H7/k6), `press` (H7/p6).

Un agujero pequeño y ajustado se escaria mejor que se fresa. Studio lo indica, y usted puede taladrarlo y luego escariarlo con un escariador de su clase.

El gemelo estima el descentramiento de la fresa, y Studio lo tiene en cuenta en las medidas que tienen un ajuste.

Varias fases

Una pieza mecanizada por dos caras necesita dos fases. Cada fase tiene sus propios elementos, e indica cómo llega la pieza a ella: volteada alrededor de X o de Y, sujeta de otra manera, con su origen de pieza encontrado de nuevo. Studio escribe un programa para cada fase y comprueba cada uno en el gemelo sobre lo que dejaron las fases anteriores. Cómo añadir fases y voltear la pieza: consulte Trabajos con varias fases.

Piezas anidadas en una placa

Se pueden cortar varias piezas de una misma placa. Studio las coloca próximas entre sí, mantiene un margen y añade puentes de sujeción para que las piezas permanezcan en la placa hasta el final.

Medidas en pulgadas y fracciones

Con unidades en pulgadas, las medidas pueden escribirse como fracciones: `1/4`, `1 1/2`, `3/16"`. Las herramientas en pulgadas están en la biblioteca de herramientas, por ejemplo una fresa plana de 1/4". Consulte Idioma y unidades.

Comprobar un programa en el gemelo

Antes de cortar, ejecute el programa en el gemelo. El gemelo corta su propia pieza e informa de colisiones, sobrecortes, vibración, la carga del husillo y el tiempo. Consulte Simulación.

> **Precaución**
>
> El gemelo comprueba el programa con el amarre que usted le dio. Prepare el gemelo como está la máquina real: el mismo bruto, el mismo utillaje, las mismas herramientas y el mismo origen de pieza.

La hoja de amarre

Una hoja de amarre dice al operador cómo preparar el trabajo: el bruto, el utillaje, las herramientas, el origen de pieza y el espacio libre en Z. Consulte Hojas de amarre.

Programas y el editor

El espacio de trabajo **Programa** (Alt+3) reúne sus programas y las piezas de ejemplo, un editor que comprueba un programa mientras usted escribe y una vista previa de lo que el programa hace a la pieza. Studio conserva una versión de un programa cada vez que usted lo guarda o lo carga en la máquina, de modo que puede ver qué ha cambiado y volver atrás.

Abrir un programa

El espacio de trabajo tiene tres partes:

- a la izquierda, **Programas** (cada programa y sus versiones) y **Piezas de ejemplo**;
- en el centro, el editor;
- a la derecha, la vista previa 3D, **Cargar y comprobar** y los hallazgos de la comprobación del programa.

Para abrir un programa:

- **Archivo > Abrir un programa…** (Ctrl+O) carga un archivo de código G en la máquina y, si el editor no tiene cambios sin guardar, también en el editor. Studio lee su archivo y nunca lo escribe.
- Soltar un archivo de código G (.nc, .gcode, .ngc, .tap, .txt, .cnc) en cualquier parte de la ventana hace lo mismo. Mientras un programa se ejecuta, Studio no abre nada y lo indica en la consola.
- **Archivo > Programas recientes** lista los ocho últimos programas abiertos desde un archivo, el más reciente primero. Uno que se haya movido o borrado desde entonces se quita de la lista al elegirlo; **Vaciar la lista** la vacía.
- **Archivo > Imprimir programa…** (Ctrl+P) imprime el programa cargado en la máquina, con las líneas numeradas, en negro sobre blanco, con su nombre, la página y la fecha arriba de cada página.
- Haga doble clic en un programa o en una versión de **Programas**: su texto pasa al editor. No pasa a la máquina hasta que usted pulse **Cargar y comprobar**.

El editor y la máquina pueden tener programas distintos. Cuando la máquina tiene un programa que usted no ha abierto en el editor, la línea de estado lo indica: Ctrl+K carga el programa del editor en la máquina y Ctrl+Mayús+R pasa el programa de la máquina al editor (antes se guardan sus ediciones como una versión).

Programas de ejemplo

Studio 1.0 incluye cuatro piezas de ejemplo: una placa de motor NEMA 17, un panel frontal de instrumento, un chip microfluídico de gotas y un panel aligerado isogrid. Llegarán más con las actualizaciones. Haga doble clic en una de ellas en **Piezas de ejemplo**, o elíjala en **Archivo > Piezas de ejemplo…**.

Al cargar un ejemplo se prepara toda la máquina para él: el bruto y el utillaje, la tabla de herramientas y la herramienta en el husillo, la referencia, el origen de pieza, el programa y, si el ejemplo lo tiene, el modelo con el que comparar.

> **Nota**
>
> Un ejemplo sustituye el bruto, la tabla de herramientas y el origen de pieza G54 que usted tenía. **Programa de demostración**, en la misma lista, carga solo el programa y deja el bruto como está.

El editor

El editor es un editor de código G:

- las palabras se colorean según su tipo: códigos G y M, ejes, palabras de arco, avance, velocidad, herramienta, comentarios, expresiones;
- el margen muestra los números de línea y las marcas: los hallazgos de la comprobación (errores en rojo, advertencias en ámbar, notas en azul), los puntos de interrupción y la línea que la máquina está ejecutando;
- tras una ejecución, un segundo margen muestra el tiempo de cada línea en segundos: en ámbar cuando no se alcanzó el avance, en rojo cuando allí ocurrió algo (un movimiento rápido contra el material, un corte con el husillo parado, una colisión, una alarma). Haga clic en él para ver las cifras de la línea en la última ejecución.

Las líneas de estado bajo el editor muestran la línea y la columna, si el texto está en la máquina, la versión, los recuentos de la comprobación y el estado del programa tras la línea en la que usted está: los códigos modales, el avance, la velocidad, la herramienta y la posición.

Expandir da al editor todo el espacio de trabajo; **Restaurar** devuelve los programas y la vista previa.

Colores y números de línea

Los colores siguen el aspecto de la ventana (claro u oscuro). Las líneas que el controlador ya ha aceptado por delante de la línea que se está ejecutando aparecen en gris.

Buscar y reemplazar

- Ctrl+F abre el cuadro de búsqueda. Intro busca la coincidencia siguiente y Mayús+Intro la anterior. La búsqueda no distingue mayúsculas de minúsculas y vuelve al principio al llegar al final.
- Ctrl+G va a un número de línea.
- Ctrl+H abre el cuadro de reemplazo bajo el cuadro de búsqueda. Intro reemplaza la coincidencia seleccionada y pasa a la siguiente; Ctrl+Intro reemplaza todas. Ctrl+Z lo deshace.

Esc cierra los cuadros.

Puntos de interrupción y pausas

Un punto de interrupción detiene el tiempo del gemelo cuando la ejecución llega a esa línea; el movimiento de la máquina no cambia. F5 continúa.

- Ctrl+B, o un clic en el margen de marcas, pone un punto de interrupción en la línea.
- **Pausar si** (Ctrl+Mayús+B) detiene el gemelo cuando ocurre algo: la herramienta empieza a cortar, un movimiento rápido corta material, una casi colisión o una colisión, una vibración, una alarma, un cambio del estado del controlador, la fuerza de corte, la carga del husillo o el error de seguimiento por encima de un límite que usted elija, o un cambio modal. Cuando el gemelo funciona en tiempo real, la comprobación de vibraciones se calcula junto a la simulación, por lo que una pausa por vibración, y las cifras de vibración en las vistas, pueden llegar un poco después del corte: normalmente unos cientos de milisegundos, y uno o dos segundos donde coinciden muchas condiciones de corte nuevas (una cajera alrededor de islas). Hasta que la comprobación ha respondido para una herramienta, el índice de vibración muestra **calculando…**; las curvas y la traza de la herramienta esperan la respuesta de cada corte, así que muestran el índice de ese corte. El informe del final de la ejecución es el mismo en ambos casos.
- El menú de la línea (clic derecho en el margen, o Mayús+F10) tiene además **Ejecutar hasta la línea …** y **Borrar todos los puntos de interrupción y condiciones**.

Los puntos de interrupción surten efecto cuando el programa está en la máquina.

La comprobación del programa

El editor comprueba el programa mientras usted escribe (un momento después de que deje de hacerlo), cuando se carga y de nuevo tras una ejecución. Tiene dos partes:

- las reglas comprueban cada línea frente a lo que aceptan el controlador y la máquina. No necesitan nada más;
- la comprobación con vista previa ejecuta el programa contra la pieza tal como está ahora. Necesita la máquina encendida y referenciada y ningún programa en ejecución. Mientras escribe, admite programas de 20 000 líneas como máximo; un programa más largo se comprueba entero una vez cargado en la máquina, y sus problemas aparecen como recuadros rojos en la vista 3D y en la pestaña **Mensajes**.

Programa > Comprobar el programa es otra comprobación: envía el programa cargado por el modo de comprobación del propio controlador, línea por línea, sin que la máquina se mueva.

Qué encuentra la comprobación

Cada hallazgo es un error, una advertencia o una nota, y se muestra bajo su línea con el número de su regla. Ejemplos:

Regla	Tipo	Qué encuentra
S01	Advertencia	Sin bloque de inicio antes del primer movimiento: el programa depende del estado del controlador
S02	Advertencia	Sin `M2` ni `M30` al final
S04	Error	Un movimiento de corte antes de cualquier `F`
S05	Advertencia	El husillo arranca sin `S`, o `S` fuera del rango del husillo
S07	Error	Un código que el controlador no acepta: expresiones, palabras `O`, subprogramas, `G41`/`G42`, `G68`
S10	Advertencia	Un movimiento rápido por debajo de la cara superior de la pieza y más profundo de lo que el programa ha cortado
S11	Error	Un arco que el controlador rechaza
S12	Advertencia	El husillo refrigerado por agua corta sin su refrigerante (`M8`)
S14	Advertencia	Una herramienta de longitud distinta de aquella con la que se fijó el origen de pieza, y nada la mide

La comprobación con vista previa añade lo que el programa haría a la pieza: movimientos rápidos contra el material, corte con el husillo parado, colisiones con el utillaje, movimientos que superan el recorrido y un husillo que funciona en seco.

Corregir un hallazgo

Algunos hallazgos traen una sugerencia, por ejemplo **Sugerencia (insertar antes): M8**.

1. Ponga el cursor en la línea, o pulse Alt+↓ y Alt+↑ para ir de un hallazgo a otro.
2. Pulse Ctrl+. (o elija **Aplicar la sugerencia** en el menú de la línea).

Resultado: el cambio se hace en un solo paso; Ctrl+Z lo deshace. Lea la sugerencia antes de aplicarla: corrige el hallazgo, no lo que usted pretendía.

Optimizar

Optimizar (Ctrl+Mayús+O) busca un programa más rápido que corte la misma pieza: cambia avances y velocidades, une líneas cortas y baja las retiradas. Ejecuta el programa original y el optimizado en dos copias ocultas de la máquina, con el mismo amarre. Necesita un bruto definido. **Optimizar** está en la edición Pro y superiores.

Si el resultado es mejor, sustituye el texto del editor (Ctrl+Z devuelve el original) y se abre la comparación. La consola indica los tiempos, la fuerza de corte más alta y el índice de vibración antes y después, y lo que cambió. Pulse **Parar optimización** o Ctrl+Mayús+O para detenerla.

Versiones

Studio guarda una versión de un programa cuando usted:

- pulsa **Guardar versión** (Ctrl+S);
- lo carga en la máquina (**Cargar y comprobar**, o **Archivo > Abrir un programa…**);
- lo optimiza, o toma un programa generado o el de la máquina en lugar de sus ediciones (antes se guardan sus ediciones).

Una versión idéntica a la última no se vuelve a guardar. Las versiones se conservan en `Documentos\Twinwright Studio\programs\<program>\`, un archivo cada una, y nunca se eliminan automáticamente. **Programas** las lista bajo cada programa, con la hora y el motivo por el que se guardó cada una.

Comparar dos versiones

Comparar (Ctrl+D) muestra, bajo el editor, la versión guardada más reciente que difiere del texto del editor: las líneas modificadas en ámbar, las añadidas en verde, las eliminadas en rojo y las líneas en las que solo cambió `F` o `S` en azul. Pulse Ctrl+D otra vez para ver una versión más antigua, Alt+↓ y Alt+↑ para ir de una diferencia a otra, y Esc para cerrar.

El encabezado cuenta las diferencias y, cuando las dos versiones se han ejecutado hasta el final en esta sesión, compara sus tiempos de ejecución.

Para comparar dos versiones guardadas, abra primero la más reciente en el editor y luego pulse Ctrl+D.

Volver a una versión

1. En **Programas**, haga doble clic en la versión. Resultado: su texto pasa al editor. Sus ediciones sin guardar se guardan antes como una versión; Ctrl+Z deshace el cambio.
2. Pulse **Cargar y comprobar** (Ctrl+K) para cargarla en la máquina.

Guardar y exportar

- **Guardar versión** (Ctrl+S) conserva una versión en la carpeta de programas de Studio.
- Ctrl+Mayús+S guarda el programa del editor como archivo donde usted elija.
- Una versión es un archivo `.nc` normal: puede copiarlo desde la carpeta de programas.

Archivo > Exportar curvas (CSV)… guarda las curvas de la ejecución (véase Análisis y diagnóstico). La hoja de amarre y el informe de inspección están en la tarjeta **Informes** de **Preparación** (véase Inspección).

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Ejecutar un programa y reanudarlo

El espacio de trabajo **Ejecución** ejecuta el programa cargado y muestra lo que hace la máquina mientras corta. Cuando una ejecución se detiene, Studio le indica el siguiente paso y puede volver a iniciar el programa desde una línea.

Cargar e iniciar

Para cargar un programa en la máquina, haga una de estas cosas:

- **Archivo > Abrir un programa...** (Ctrl+O): elija un archivo de código G. Studio lo lee; nunca escribe su archivo.
- **Archivo > Piezas de ejemplo...**: elija un ejemplo. Studio también prepara para él el bruto, las herramientas y el origen de pieza.
- En el espacio de trabajo **Programa**, **Cargar y comprobar** (Ctrl+K) carga el programa que está en el editor.

La barra de la máquina muestra entonces el nombre del programa, su número de líneas y cuánto tarda aproximadamente.

Para iniciarlo, pulse **Inicio de ciclo** (F5) en la barra de la máquina. **Inicio de ciclo** aparece relleno cuando el programa está cargado y la máquina está en reposo y referenciada. Si la comprobación del programa encontró problemas, la consola lo indica al iniciarse el programa.

Antes de ejecutar un programa en una máquina real, compruébelo: **Programa > Comprobar el programa** lo envía por el modo de comprobación del controlador. El controlador lee cada línea y la máquina no se mueve.

Observar una ejecución

Mientras se ejecuta un programa:

- la vista 3D muestra la herramienta, la traza de su punta y la pieza a medida que se corta. **Detalle de herramienta** muestra la herramienta en el corte;
- la pestaña **Programa** del panel inferior muestra el listado: la línea que se está ejecutando está marcada, las líneas ya ejecutadas son más claras y una barra muestra el tiempo de cada línea. **Seguir la línea** mantiene la línea a la vista;
- **Estructura del programa**, junto al listado, divide el programa en sus secciones y capas (a partir de los comentarios del programa) con sus tiempos;
- la columna derecha muestra las lecturas.

Cuando el programa termina, la pestaña **Mensajes** indica cuántas líneas se ejecutaron y cuánto duró la ejecución en la simulación.

Progreso y tiempo restante

La barra de la máquina muestra la línea que se está ejecutando sobre el total de líneas, el porcentaje de líneas hechas, y **transcurrido** y **restante**:

- **transcurrido** es el tiempo de máquina hasta ahora, en el tiempo de la simulación;
- **restante** es la estimación del resto, a partir de los avances y los movimientos rápidos del programa.

Tras una alarma, la barra muestra la línea en la que se detuvo el programa.

Las lecturas

La columna derecha de **Ejecución**, de arriba abajo:

- **Posición**: cada eje en coordenadas de pieza y de máquina, con una marca cuando el eje está referenciado. Debajo, la línea modal: los códigos G vigentes, la herramienta, el avance, la velocidad del husillo y los códigos M. En una alarma, el husillo y el refrigerante se muestran como los tiene la máquina, y los valores del programa están en la descripción emergente;
- **Husillo y avance**: la velocidad del husillo (el arco es la velocidad y la marca, la velocidad solicitada), la carga del husillo (verde hasta 80 %, rojo por encima de 100 %) y el avance;
- **Corte** (nivel Ingeniero y superiores): las estimaciones del gemelo de la fuerza de corte, la potencia, el índice de vibración y el avance por diente;
- **Ajustes (override)**, **Panel de mando**, **Manual** y **Herramientas y origen** (nivel Operador y superiores).

Haga clic en el título de una sección para plegarla. En el modo taller, **Posición** muestra además **Restante**: la distancia que queda en la línea actual, para cada eje.

Retención de avance, parada y overrides

- **Retención de avance** (F6 o Espacio) frena los ejes hasta detenerlos; el husillo sigue girando. **Inicio de ciclo** continúa.
- **Parar** (F7) primero hace una retención de avance y reinicia el controlador cuando la máquina se ha detenido. Púlselo otra vez para reiniciar de inmediato.
- **Reset** (F12) detiene el programa de inmediato.

Los overrides cambian el avance del programa y la velocidad del husillo mientras se ejecuta, de 10 % a 200 %:

Override	Teclas	En pantalla
Avance	Ctrl+↑, Ctrl+↓: ±10 %	**Ajustes (override)**: arrastre el mando de ajuste; Ctrl para pasos de 1 %
Husillo	Ctrl+→, Ctrl+←: ±10 %	**Ajustes (override)**: arrastre el mando de ajuste
Rápido	**Máquina > Ajustes (override)**: 25 %, 50 %, 100 %	**Ajustes (override)**: 100, 50, 25

Ctrl+0, o un doble clic en un mando de ajuste, devuelve el avance y el husillo a 100 %.

Bloque a bloque

Bloque a bloque, en la barra de la máquina, detiene el programa después de cada línea. Pulse **Inicio de ciclo**, o **Programa > Bloque siguiente**, para ejecutar la línea siguiente. Puede activarlo antes de iniciar o mientras el programa se ejecuta; haga clic en él otra vez para continuar sin paradas.

Desplazamiento manual (jog) y volante

Haga el desplazamiento manual en **Ejecución**, en **Manual** (empieza plegado), o con las teclas (véase Las teclas de la máquina).

- **Incremento**: 0.01, 0.1, 1, 10 o 50 mm por pulsación, o **Jog continuo**: el eje se mueve mientras usted mantiene pulsada la tecla.
- **Velocidad**: 100, 300, 1500 o 3000 mm/min.
- La botonera: X e Y en cruz, Z a su lado y **Parar desplazamiento manual** (Esc) en el centro.
- **Volante**: elija **X**, **Y** o **Z** y el incremento (**×1**, **×10**, **×100**: 0.001, 0.01 o 0.1 mm por muesca) y luego arrastre el dial o gire la rueda del ratón sobre él.

Solo puede hacer el desplazamiento manual cuando la máquina está encendida, en reposo y sin alarma, y no hay ningún programa en ejecución. Un desplazamiento mantenido se detiene cuando usted suelta la tecla o cuando la ventana pierde el foco.

Herramientas > Ajustes... > Panel y dispositivos > Volante electrónico define el comportamiento del volante: **Velocidad** detiene el eje cuando se detiene la rueda; **Trayectoria** mueve cada muesca.

Alarmas y qué hacer

En una alarma, el estado de la barra de la máquina se vuelve rojo y la barra muestra:

- la alarma y lo que significa, por ejemplo **ALARM:17 Un accionamiento de motor informa de un fallo**;
- **Siguiente paso:** qué hacer ahora, con la tecla que lo hace;
- **Todos los pasos**, cuando hay más de uno.

Haga clic en la tecla, o use **Máquina > Realizar el paso siguiente**, para realizar el siguiente paso. Después de cada paso, Studio vuelve a calcular los pasos a partir del estado de la máquina.

Las otras señales de una alarma: una línea bajo **Posición** cuando ya no se puede confiar en la posición, la línea roja del listado donde se detuvo el programa y una etiqueta en el accionamiento en la vista 3D.

Las alarmas, sus causas y lo que las borra: Alarmas.

Los pasos para volver al trabajo

Todos los pasos abre los pasos para volver al trabajo: la causa y después los pasos en orden, cada uno con su tecla y el motivo por el que hace falta. El paso que toca ahora está marcado. Un clic hace solo el siguiente paso.

Por ejemplo, tras una alarma de accionamiento (ALARM:17):

1. **Pulsar la parada de emergencia y cortar la alimentación de los accionamientos** (F1): la alarma de un accionamiento solo se borra cuando este pierde la alimentación.
2. **Liberar la parada de emergencia** (F1).
3. **Encender** (F2): los accionamientos tienen alimentación de nuevo.
4. **Reiniciar** (F12): el controlador mantiene ALARM:17 después de que los accionamientos se recuperan.
5. **Referenciar** (F3): la alimentación estuvo cortada, así que no se debe confiar en la posición.

Cuando el fallo se inyectó en **Diagnóstico**, la causa lo indica, con la hora.

Cuando una alarma detuvo un programa, los pasos terminan con la línea desde la que volver a empezar, por ejemplo el inicio de la capa en la que se detuvo.

Reanudar desde una línea

Studio puede iniciar un programa desde cualquier línea. Escribe un bloque de inicio que deja la máquina como la tendría el programa en esa línea y después ejecuta el resto.

1. Detenga el programa. La máquina debe estar encendida y no debe haber ningún programa en ejecución.
2. En la pestaña **Programa** del panel inferior, haga clic derecho en la línea y elija **Iniciar desde la línea ...**, y luego **Aceptar**. En el editor, haga clic derecho en el margen de la línea o pulse Mayús+F10. Resultado: Studio carga "*nombre* (desde la línea *n*)". No lo inicia.
3. Pulse **Inicio de ciclo**.

El bloque de inicio:

1. sube Z hasta arriba;
2. establece las unidades, el plano, el modo de distancia y el decalaje de origen que tenía el programa en esa línea;
3. cambia la herramienta, si el programa la había cambiado antes de esa línea;
4. enciende el refrigerante y el husillo a la velocidad del programa, y espera a que el husillo alcance la velocidad;
5. se coloca sobre el punto de inicio a una altura segura, baja hasta 2 mm por encima de él y luego desciende con avance de trabajo;
6. continúa con el programa desde esa línea.

La comprobación con vista previa comprueba luego el resto frente a la pieza tal como está.

> **Precaución**
>
> Studio no puede seguir algunas líneas anteriores al inicio: parámetros, expresiones, subprogramas, `G10`, `G28`, `G30` y `G92`. Las notas de la carga las enumeran. Compruebe el estado de la máquina antes de pulsar **Inicio de ciclo**.

Probar primero el inicio en una copia

Cuando una alarma detuvo un programa, los pasos terminan con **Vista previa del inicio…**. Abre el bloque de inicio y sus notas.

1. Haga clic en **Probar en una copia**. Resultado: el inicio se ejecuta en una copia oculta de la máquina, a partir de la pieza tal como está, y Studio informa de su tiempo, de cualquier colisión, del índice de vibración más alto y de cuánto material quitó.
2. Si le parece correcto, siga los pasos para volver al trabajo y luego inicie desde la línea.

Ejecutar con la velocidad de la simulación

El gemelo funciona en tiempo real. **Simulación > Velocidad** lo ejecuta de ×0.5 a ×50 el tiempo real. El reloj de la barra de la máquina muestra el tiempo de la simulación, la velocidad que usted eligió y la velocidad que alcanza el equipo.

Simulación > Pausar la simulación, o el reloj de arena de la barra de la máquina, detiene el tiempo del gemelo: nada cambia mientras está detenido. **Inicio de ciclo** continúa. Los puntos de interrupción y las condiciones de pausa del editor detienen el tiempo del gemelo de la misma manera (véase Programas y el editor).

Pausar la simulación no es una retención de avance: el estado de la máquina queda congelado, no detenido.

Análisis y diagnóstico

Análisis repasa una ejecución: adónde fue el tiempo, las estimaciones del gemelo de fuerza y vibración, y cómo se compara la pieza cortada con el modelo de diseño. **Diagnóstico** muestra la cadena de alimentación y las señales de la máquina, y le permite inyectar fallos en el gemelo para practicar cómo encontrarlos.

Examinar una ejecución

Análisis (Alt+5, nivel Ingeniero y superiores) se rellena en cuanto se ha ejecutado un programa.

- **Ejecuciones**, a la izquierda, lista las ejecuciones de esta sesión: el programa, cómo terminó, su tiempo de máquina y cuándo empezó. Las ejecuciones que terminaron en una alarma o un error están en rojo.
- En el centro están las curvas y, bajo ellas, dos pestañas: **Perfil por línea** y **Lóbulos de estabilidad**.
- **Última ejecución**, a la derecha, resume la última ejecución: hasta dónde llegó, las líneas que más tiempo tomaron, la fuerza de corte más alta y el índice de vibración más alto con sus líneas, el material quitado, las líneas en las que no se alcanzó el avance y los eventos.

Las curvas, el perfil y el resumen son los de la última ejecución.

Perfil por línea lista las líneas que tomaron tiempo, de la más larga a la más corta: el tiempo y su porcentaje, el avance solicitado y el avance alcanzado, por qué no se alcanzó, la fuerza de corte, el índice de vibración, el material quitado y los eventos. Haga doble clic en una línea, o pulse Intro sobre ella, para ir a ella en el editor de **Programa**.

Por qué una línea no alcanzó su avance:

- la velocidad máxima de los ejes;
- esquinas demasiado próximas entre sí para acelerar entre ellas;
- el límite de cuerda del arco;
- la planificación del controlador, o un retraso del programa de envío o de la lectura del controlador;
- un override de avance por debajo de 100 %, o una retención de avance.

Tiempo: estimado y real

Antes de una ejecución, Studio estima el tiempo de máquina a partir de los avances y los movimientos rápidos del programa: la barra de la máquina muestra **aprox.** ese tiempo y la **Estructura del programa** del panel inferior lo muestra para cada sección. Durante y después de una ejecución, los tiempos son el tiempo de máquina del gemelo, con la aceleración y la planificación del controlador incluidas: la barra de la máquina muestra el tiempo transcurrido y el restante, **Estructura del programa** el tiempo de toda la ejecución y **Perfil por línea** el tiempo de cada línea.

Todos los tiempos son tiempo de máquina en la simulación, no el tiempo de su reloj.

Fuerza, vibración y espesor de viruta

Las curvas muestran las lecturas de la ejecución durante los últimos 10, 30 o 120 s: el avance, la velocidad real, la velocidad y la carga del husillo, la fuerza de corte, el par y la potencia, el espesor de viruta, el índice de vibración y los errores de seguimiento. Haga clic en el nombre de una curva para mostrarla u ocultarla.

El índice de vibración es la profundidad de corte dividida por la profundidad estable a la velocidad de husillo en uso:

Índice	Significa
Menos de 0.8	Estable
0.8–1	Cerca del límite
Más de 1	Hay vibración

Lóbulos de estabilidad dibuja la profundidad de corte estable frente a la velocidad del husillo, para el último corte (o, si aún no ha habido ningún corte, para una ranura completa de 1 mm de profundidad a lo largo de X). Haga clic en **Calcular los lóbulos de estabilidad**. Un punto rojo marca el corte tal como fue y una línea discontinua verde, la velocidad más estable.

Ver > Coloración colorea la traza de la punta de la herramienta según la velocidad de avance, la fuerza de corte, el espesor de viruta, el índice de vibración o la línea del programa, y la pieza según el material y la profundidad, la desviación respecto al modelo o la línea del programa.

Son estimaciones del gemelo obtenidas de un modelo físico (véase Modelos del gemelo).

Comparar con el modelo de diseño

Para ver dónde difiere la pieza cortada del diseño:

1. En **Preparación**, paso **Amarre**, haga clic en **STL del modelo (comparar)…** y elija el archivo STL del modelo (en mm, con su origen en el cero del programa).
2. Ejecute el programa.
3. Elija **Ver > Coloración > Pieza: desviación respecto al modelo**.

Resultado: la pieza se colorea según lo lejos que está cada punto del modelo: dentro de ±20 µm cuenta como bueno y los demás colores muestran material sin retirar y sobrecortes. **Dejar de comparar** quita el modelo. Las piezas de ejemplo cargan su propio modelo.

Comparar dos ejecuciones

Análisis muestra la última ejecución. Para comparar dos ejecuciones de un programa:

- en el editor de **Programa**, **Comparar** (Ctrl+D) da los tiempos de ejecución de ambas versiones y el tiempo de las líneas que cambiaron, cuando las dos se han ejecutado hasta el final en esta sesión (véase Comparar dos versiones);
- el asistente de IA puede dibujar dos ejecuciones una junto a otra (véase El asistente de IA).

Exportar

Exportar curvas (CSV)… (en **Última ejecución**, o **Archivo > Exportar curvas (CSV)…**) guarda aproximadamente los últimos 160 s de las curvas como archivo CSV: una columna de tiempo en segundos y después una columna por cada curva, con su unidad en el encabezado y los valores en las unidades de visualización.

Exportar pieza… en **Preparación**, o **Archivo > Exportar pieza…**, guarda la pieza cortada como archivo STL (en coordenadas de programa, de bruto o de máquina) o como mapa de alturas.

El espacio de trabajo Diagnóstico

Diagnóstico (Alt+6, nivel Técnico y superiores) muestra la cadena de alimentación de la máquina como un diagrama: la energía fluye de izquierda a derecha. Las líneas continuas conducen energía, en verde cuando tienen tensión; las líneas discontinuas son señales. Cada componente está en verde cuando está bien, en gris cuando está apagado, en ámbar si hay una advertencia y en rojo si hay una alarma o un fallo.

Haga clic en un componente para ver sus detalles en la columna de la derecha. Bajo el diagrama están las entradas del controlador, la inyección de fallos y las alarmas de los accionamientos.

La cadena de alimentación

La cadena del MC-1, de la red eléctrica al husillo:

- **Interruptor principal QF1** y la **Fuente de 24 V** para el controlador;
- el **Controlador**: su estado o su alarma;
- **Parada de emergencia SB0** y **KM1**, el contactor que esta abre;
- la **Fuente de 48 V** para los accionamientos;
- los accionamientos **X**, **Y** y **Z** y sus ejes;
- el **Variador de frecuencia**, el **Husillo** y la **Bomba de refrigeración**.

Las entradas del controlador

Una lámpara para cada entrada: los finales de carrera (**Final de carrera X−**, **Final de carrera X+**, **Final de carrera Y−**, **Final de carrera Y+**, **Final de carrera Z**), el **Sensor de longitud de herramienta**, la **Parada de emergencia** y la **Alarma del accionamiento**. Una lámpara está en gris cuando la entrada está desactivada y en ámbar o rojo cuando está activada.

La lámpara **Sensor de longitud de herramienta** es la entrada de palpación del controlador: también se enciende cuando el palpador hace contacto.

Los accionamientos

Cada accionamiento muestra **Listo** y su error de seguimiento, o **Deshabilitado**, **Alarma**, **Arrancando** o **Apagado**. Los detalles dan la alimentación de los accionamientos y el error de seguimiento de cada eje, con su valor más alto.

El variador del husillo (variador de frecuencia)

El **Variador de frecuencia** muestra su frecuencia y su corriente de salida, o su fallo: **OC** (sobrecorriente), **OL** (sobrecarga) o **LU** (subtensión). Los detalles dan la velocidad del husillo y la velocidad solicitada, la carga y la bomba. Un husillo que funcionó sin su refrigerante se indica entre las advertencias, con el tiempo que funcionó en seco.

Inyectar fallos en el gemelo

La inyección de fallos le permite practicar cómo encontrar y resolver fallos en el gemelo; se cambia la máquina de su ventana, no una máquina real.

- **Cables**: ponga cualquiera de los cables de la máquina (los finales de carrera, el sensor de longitud de herramienta, la cadena de alarma de los accionamientos, la línea de la parada de emergencia, el circuito de KM1, el bus RS485, el relé de fallo del variador de frecuencia) en **Circuito abierto**, **Cortocircuito**, **Intermitente** o **Interferencias** (una única interrupción de 50 µs). **Normal** deja el cable en orden.
- **Alarma del accionamiento**: **X**, **Y** o **Z** hace que ese accionamiento informe de un fallo. La alarma de un accionamiento solo se borra cuando este pierde la alimentación: siga los pasos para volver al trabajo de la barra de la máquina.

Cada fallo inyectado se indica entre las advertencias, se escribe en la consola y se registra en el registro de auditoría. Tras una alarma de accionamiento, la causa que muestra la barra de la máquina nombra el fallo que usted inyectó y la hora.

Alarmas: cada una y su recuperación

Todas las alarmas, lo que significan y sus causas: Alarmas. La barra de la máquina da los pasos para volver al trabajo para la alarma que usted tenga (véase Los pasos para volver al trabajo):

Alarma	Pasos para volver al trabajo
Fallo de un accionamiento (ALARM:17 con un accionamiento en alarma)	Pulsar la parada de emergencia, liberarla, **Encender**, **Reiniciar**, **Referenciar**
Parada de emergencia pulsada (ALARM:10)	Liberar la parada de emergencia, **Encender**, **Reiniciar**, **Referenciar**

Alarma	Pasos para volver al trabajo
ALARM:17 con los accionamientos recuperados	**Reiniciar**, **Referenciar**
Falta la referencia (ALARM:11)	**Referenciar**
Límite físico (ALARM:1)	**Reiniciar**, **Desbloquear**, sacar el eje del final de carrera con un desplazamiento manual, **Referenciar**
ALARM:3, 8, 9	**Reiniciar**, **Referenciar**
Cualquier otra alarma	**Reiniciar**, **Desbloquear**

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Ejecutar un programa en el gemelo

Borrador (ronda 14): el texto se está redactando; las secciones aún pueden crecer.

Qué es el gemelo

La máquina de la ventana es un gemelo digital del MC-1: en él se ejecuta el núcleo propio de Twinwright, el mismo código que en la máquina, y un modelo físico representa los motores, los accionamientos, el husillo, los interruptores y el corte. Un programa se ejecuta en el gemelo como lo haría en la máquina, y el gemelo corta su propia pieza. Todo número que da sobre el mundo físico es una estimación.

Antes de ejecutar

El gemelo comprueba un programa frente al amarre que usted le da, así que prepárelo como está la máquina real:

- la máquina (**Archivo > Cambiar de máquina…**);
- el bruto y cómo se sujeta (el espacio de trabajo **Preparación**);
- las herramientas (la **Tabla de herramientas** del panel inferior);
- el origen de pieza (**Máquina > Origen de pieza**).

Archivo > Piezas de ejemplo… establece todo esto y carga un programa de una sola vez.

> **Precaución**
>
> Una comprobación vale lo que valga el amarre: un bruto, un utillaje, una herramienta o un origen de pieza distintos en la máquina real no son lo que el gemelo comprobó.

Ejecutar un programa

1. Abra el programa: **Archivo > Abrir un programa…**
2. Compruébelo primero: **Programa > Comprobar el programa** lo pasa por el modo de comprobación del controlador, de modo que el controlador lee cada línea y la máquina no se mueve.
3. Ejecútelo: **Inicio de ciclo** (F5). **Retención de avance** (F6) lo pausa, **Inicio de ciclo** continúa y **Parar** (F7) lo detiene. **Bloque a bloque** lo ejecuta línea por línea y **Bloque siguiente** ejecuta la línea siguiente.

El gemelo funciona en tiempo real, salvo que usted elija otra cosa: **Simulación > Velocidad** lo ejecuta de la mitad a 200 veces el tiempo real, o a **Máxima** —tan rápido como pueda este equipo—, y **Simulación > Pausar la simulación** detiene el tiempo del gemelo (el estado de la máquina no cambia mientras está detenido). Un clic en el reloj, a la derecha de la barra de la máquina, abre la misma lista.

A partir de ×100, en **Máxima** y mientras calcula hasta el final, el gemelo funciona en **Ligero** (la etiqueta junto al reloj): la comprobación de vibraciones y las fuerzas pueden llegar a las curvas algo más tarde, y la imagen usa celdas más gruesas y una traza más fina. Lo que hace el programa —el material arrancado, las colisiones, los límites de software, las alarmas— se juzga como a ×1, y el resultado y el informe son los mismos. **Ajustes > General > Precisión completa también a alta velocidad (más lento)** desactiva Ligero.

Cuando el equipo no da abasto, el reloj indica en ámbar a qué velocidad funciona realmente; su descripción emergente nombra la parte del gemelo más ocupada (por ejemplo, el arranque de material en celdas finas) y qué hacer.

Simulación > Velocidad > Calcular hasta el final (sin reproducción) ejecuta el programa cargado hasta el final sin mostrarlo fotograma a fotograma: la barra de la máquina muestra hasta dónde ha llegado y el tiempo restante, la vista 3D se queda en el inicio y cambia cuando termina, y la última ejecución del espacio de trabajo Análisis se rellena como de costumbre. **Parar**, junto al programa, lo detiene donde está. Conectado a una máquina real (modo sombra) solo es posible ×1: el gemelo sigue el tiempo de la máquina.

Qué comprueba el gemelo

Lo que se deduce del programa y de la geometría del amarre:

- las alarmas y los errores del propio controlador, tal como los da el controlador de la máquina: límites de software, finales de carrera, un movimiento de palpación que no toca nada;
- colisiones del portaherramientas, la tuerca de la pinza y el husillo con el bruto, el utillaje y la mesa, y casi colisiones;
- corte en un movimiento rápido (`G0`) y corte con el husillo parado;
- corte que atraviesa el fondo del bruto y llega al utillaje o a la mesa;
- una herramienta que se atascaría o se rompería: un avance por diente demasiado alto para la herramienta, una tensión de flexión en la raíz de los filos por encima de lo que soporta la herramienta;
- el espacio libre en Z: si cada herramienta, y el palpador, pasa por encima del amarre con Z arriba (véase Hardware del MC-1);
- con el modelo de diseño de la pieza en el amarre: material sobrante en la pieza y cortes demasiado profundos (**Ver > Coloración > Pieza: desviación respecto al modelo**).

Los hallazgos aparecen en la pestaña **Mensajes** del panel inferior a medida que ocurren.

Qué estima el gemelo

Lo que calcula el modelo físico:

- las fuerzas de corte, el par y la potencia, y la carga del husillo: el husillo se frena bajo carga como lo haría uno real, y demasiada carga hace saltar su variador;
- cómo ceden la herramienta y la máquina bajo el corte, en la forma cortada;
- la vibración: el índice a lo largo de la trayectoria (**Ver > Coloración > Traza: índice de vibración**) y las velocidades de husillo en las que el corte es estable (el espacio de trabajo **Análisis**: **Calcular los lóbulos de estabilidad**);
- el descentramiento de la fresa: las ranuras y los agujeros salen más anchos en esa medida;
- el tiempo de máquina, a partir de la planificación propia del controlador y del movimiento de los accionamientos;
- las crestas que una fresa de punta esférica deja entre sus pasadas (su forma, no la rugosidad de la superficie).

La traza también puede colorearse según la velocidad de avance, la fuerza de corte o el espesor de viruta (**Ver > Coloración**), y la pestaña **Curvas** del panel inferior representa la ejecución a lo largo del tiempo.

> **Nota**
>
> Las estimaciones usan parámetros típicos — datos de corte publicados, herramientas de carburo afiladas, los valores de diseño de la máquina — hasta que se mida su máquina. Úselas para comparar programas y encontrar problemas antes de cortar, no como una medición de la pieza.

Qué no modela el gemelo

- la textura y la rugosidad de la superficie;
- el filo recrecido, las virutas y su evacuación;
- las paredes delgadas que se flexionan bajo el corte;
- el calor propio de la pieza;
- el desgaste de la herramienta: el corte es siempre el de una herramienta afilada. El gemelo lleva una estimación del desgaste de cada herramienta —cuánto tiempo estuvieron sus filos en el material, qué longitud cortaron, qué parte de una vida útil estimada se ha consumido (ley de Taylor; los valores de la mayoría de los materiales son estimaciones)— y la informa (tool_wear de machine_status por MCP), pero no cambia el corte; una herramienta nueva de la misma especificación empieza de cero (setup_tool con new). Las herramientas que cortan aluminio suelen cambiarse antes, porque el aluminio se pega al filo y la pieza muestra rebabas o un acabado peor;
- lo que ocurre después de que se rompa una herramienta;
- los detalles más finos que la cuadrícula de la comprobación del programa: antes de que se ejecute nada, la comprobación del programa examina la pieza en celdas de unos 0.1 mm, más finas cuando el programa tiene una herramienta de menos de 2 mm de diámetro: una vigésima parte de su diámetro, hasta 0.05 mm (más gruesas en brutos de más de 100 × 100 mm: un millón de celdas como máximo, cuatro millones con una herramienta así), así que un movimiento rápido a través de una pared más delgada, o una herramienta que roza una pared por menos, puede pasar la comprobación sin que se note. Ejecute el programa en el gemelo para verlo: el corte del propio gemelo trabaja en celdas de 0.02 mm.

Plásticos y microherramientas

Al cortar PMMA (acrílico) con herramientas de 0.1–0.5 mm (canales microfluídicos, por ejemplo), el gemelo además deja fuera:

- la temperatura del corte: el PMMA se ablanda al acercarse a su transición vítrea (unos 105–115 °C), y a partir de unos 70 °C sus virutas se sueldan y se embadurnan en la herramienta y en el canal. Mantenga el avance por diente y la velocidad en el rango que da el fabricante de la herramienta;
- el chorro de aire o el refrigerante, y lo que hacen con el calor y las virutas;
- las rebabas a lo largo de los bordes de canales y cajeras;
- las paredes delgadas: en PMMA, las paredes de menos de unos 0.15 mm de espesor se doblan o se rompen bajo el corte; el gemelo corta la forma que dejan la trayectoria de la herramienta y su flexión, no la deformación de la propia pared;
- la amortiguación del propio corte: una herramienta tan pequeña se flexiona a decenas de kilohercios y corta despacio, y su flanco, al rozar la superficie cortada, amortigua su vibración. El gemelo la deja fuera, así que su índice de vibración para esas herramientas queda del lado seguro.

Los modelos y lo que suponen: Modelos del gemelo.

Los resultados y el informe

- La pieza de la vista 3D es lo que cortó el gemelo; su coloración muestra la profundidad del corte o, con el modelo de diseño, la desviación respecto a él.
- La pestaña **Mensajes** del panel inferior enumera los hallazgos de las comprobaciones; su pestaña **Curvas**, las fuerzas, la carga y las velocidades a lo largo del tiempo. **Archivo > Exportar curvas (CSV)…** las guarda.
- Un cliente de IA lee los mismos resultados mediante las herramientas MCP de Studio: véase la referencia de Herramientas MCP.

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Inspección, informes y hojas de amarre

Después de mecanizar una pieza, el palpador la mide con respecto al diseño. Studio escribe lo que midió, con la estimación del gemelo al lado, en un informe de inspección. Antes de mecanizar una pieza, una hoja de amarre le dice al operario cómo preparar el trabajo. Ambos están en la edición Pro.

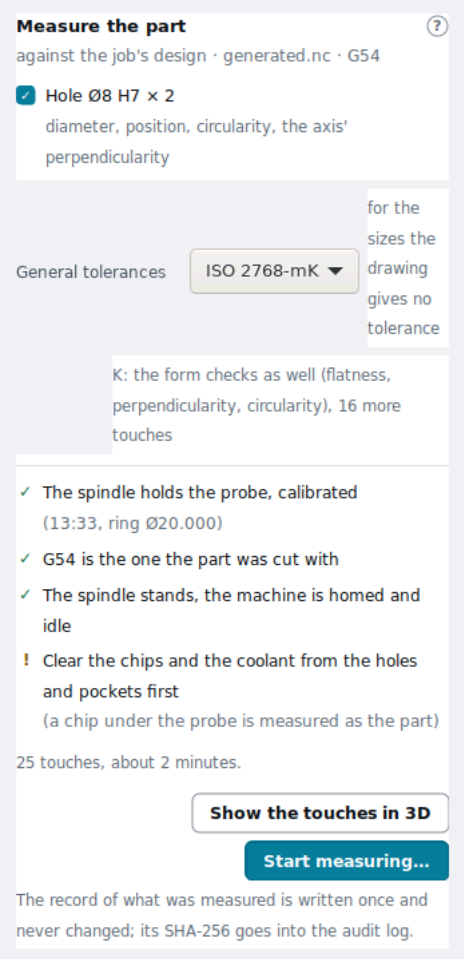
Medir una pieza con el palpador

Mida una pieza donde se mecanizó, antes de retirarla de la máquina. La tarjeta **Medir la pieza** está en **Preparación**, en el paso **Cero**, debajo de la tarjeta del palpador. Compara la pieza con el diseño de su trabajo CAM: el tamaño, el ajuste y la tolerancia de cada elemento. En la versión 1.0 la tarjeta mide en el gemelo, igual que la tarjeta del palpador. Cuando el gemelo sigue a una máquina real, mida allí con el control de la propia máquina.

Necesita:

- La pieza, ya mecanizada y todavía sujeta tal como estaba al mecanizarla. No mueva el origen de pieza: las posiciones se miden desde él.
- El palpador en el husillo, calibrado en su anillo patrón. Vea Palpado.
- El trabajo CAM del programa: un programa hecho en **CAM** o por el asistente de IA.

> **Nota**
>
> La medición está en la edición Pro. En las demás ediciones la tarjeta aparece con una etiqueta **Pro**, y **Iniciar medición…** dice cómo probarla. Los registros y los informes hechos antes se abren en todas las ediciones.



Para medir:

1. En **Preparación**, haga clic en el paso **Cero**. Resultado: la tarjeta dice con qué compara la pieza: el programa y el decalaje de origen con el que se mecanizó, por ejemplo G54.
2. Elija los elementos que va a medir. Al principio están todos elegidos. Resultado: cada elemento dice qué mide el palpador en él. Un elemento que el palpador no puede medir aparece en gris y dice por qué.
3. En **Tolerancias generales**, elija la clase para las medidas que no tienen tolerancia propia. Añada H, K o L para las comprobaciones geométricas (consulte Tolerancias). Resultado: la tarjeta dice cuántos toques hace y aproximadamente cuánto tardan.
4. Lea las comprobaciones: el palpador en el husillo y calibrado; el origen de pieza, el mismo con el que se mecanizó la pieza; el husillo parado; la máquina referenciada y en reposo.
5. Retire las virutas y el refrigerante de los agujeros y las cajeras. Una viruta bajo el palpador se mide como si fuera la pieza.
6. Para ver dónde tocará el palpador, haga clic en **Mostrar los toques en 3D**. Resultado: la vista 3D muestra los toques laterales y los toques hacia abajo.

7. Haga clic en **Iniciar medición...**. Resultado: el palpador toca cada elemento, y la tarjeta dice qué está midiendo.

> ⚠ **Advertencia**
>
> El palpador se mueve a velocidades de avance cerca de la pieza. Mantenga las manos y las herramientas alejadas de la máquina mientras mide.

Qué se mide:

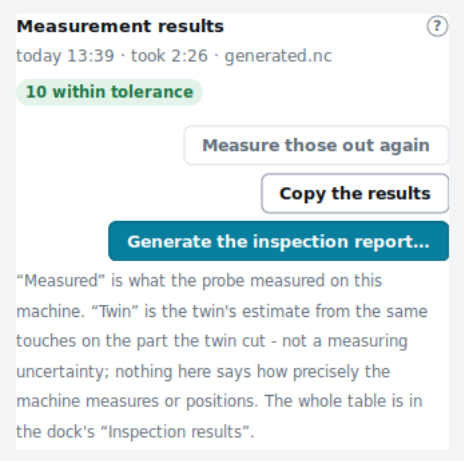
Elemento	Comprobaciones
Cajera, contorno	El tamaño y la posición de sus paredes, la profundidad de su fondo
Agujero, saliente	Diámetro y posición
Ranura	Anchura y profundidad
Planeado	Profundidad
Chaflán	Anchura, profundidad y ángulo
Ranura en V (grabado o tallado en V con una fresa en V)	Anchura, profundidad, ángulo y desplazamiento respecto a su línea
Grabado con una fresa plana, rosca, superficie 3D	No se miden

Las comprobaciones geométricas añaden planitud, perpendicularidad, redondez y rectitud donde corresponda (vea Forma).

Cuando el palpador termina, la tarjeta **Resultados de la medición** ocupa el lugar de la tarjeta de medición. Muestra:

- cuándo se midió la pieza, cuánto duró y el programa;
- cuántas comprobaciones están dentro de tolerancia, cuántas fuera de tolerancia y cuántas no se evalúan: **no evaluado en el gemelo** (allí sus celdas son mayores que una cuarta parte de la tolerancia), **sin tolerancia** o **no medido**;
- cada comprobación fuera de tolerancia, con su valor medido, su límite y cuánto excede o queda por debajo;
- para el primer elemento fuera de tolerancia, la causa más probable y qué hacer a continuación.

En la versión 1.0 el palpador mide la pieza que mecanizó el gemelo, así que la causa sale solo del gemelo. Si la pieza del propio gemelo está fuera de tolerancia en los mismos toques, es el programa el que la mecaniza así: cámbielo en **CAM**. Si no, el valor medido y la estimación del gemelo solo difieren en lo que el gemelo modela del palpador (el radio efectivo de la bola y, en un palpador típico, su precurso, que una calibración en el anillo patrón elimina) y en el tamaño de las celdas del gemelo donde toca el palpador. Calibre el palpador en el anillo patrón y vuelva a medir. Las causas son sugerencias, no un diagnóstico.



Los botones de la tarjeta:

- **Medir de nuevo las que están fuera**: mide otra vez las comprobaciones fuera de tolerancia, por ejemplo después de volver a calibrar el palpador.
- **Copiar los resultados**: copia la tabla como texto, una columna por tabulación, para una hoja de cálculo.
- **Generar el informe de inspección...**: escribe el informe (consulte El informe de inspección).

La tabla completa está en la pestaña **Resultados de la inspección** del panel inferior. Cada fila es una comprobación, con su valor de **Diseño**, su **Tolerancia**, lo **Medido**, la **Desviación**, la estimación del **Gemelo** y el veredicto: ✓, o ✗ con cuánto se sale. Las filas fuera de tolerancia están en rojo, y la vista 3D marca los resultados en la pieza.

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	✓

Cada inspección se guarda como un registro. El registro se escribe una sola vez y nunca se modifica, y su SHA-256 pasa al registro de auditoría (consulte Comprobar una copia con el registro de auditoría).

El asistente de IA mide de la misma manera, con la herramienta `inspect_part`. Pida, por ejemplo: "Inspeccione la pieza de bracket.nc con el palpador".

1. El asistente dice qué va a medir y cuántos toques y segundos añaden las comprobaciones geométricas.
2. Confirme antes de que el palpador se mueva. Resultado: la respuesta enumera cada comprobación con su valor medido, la estimación del gemelo y el veredicto.

Tolerancias

Cada comprobación se juzga según una tolerancia. La tolerancia propia de una medida, o su ajuste, viene del trabajo CAM, tal como se escribió en el campo de la medida (por ejemplo 8 H7). Las demás medidas toman la tolerancia general.

Tolerancias generales (ISO 2768-1)

ISO 2768-1 tiene cuatro clases: f (fina), m (media), c (gruesa) y v (muy gruesa). Una medida sin tolerancia propia toma la tolerancia de la clase para su intervalo de medidas. Elija la clase en la tarjeta, en **Tolerancias generales**. O pídala al asistente, por ejemplo: "Inspecciónela según ISO 2768-m".

Forma: planitud, perpendicularidad, redondez, rectitud (ISO 2768-2)

Añada H, K o L después de la clase, por ejemplo "mK". Entonces el palpador comprueba también la forma:

Comprobación	Cómo se mide
Planitud	Un fondo tocado en una cuadrícula de 3 × 3
Perpendicularidad	Una pared, o el eje de un agujero, tocados de nuevo a una segunda profundidad
Redondez	Un agujero o saliente tocado en 8 puntos
Rectitud	Una pared larga tocada en 3 a 8 puntos

La tarjeta, o el asistente, dice cuántos toques más requieren las comprobaciones geométricas antes de que el palpador se mueva.

> **Nota**
>
> Un palpador que no está calibrado en cada dirección tiene su propia lobulación. El informe indica de qué tamaño la considera el modelo de palpador del gemelo, junto a cada redondez.

Ajustes (ISO 286)

Una medida escrita con su ajuste, por ejemplo Ø8 H7, se juzga según la zona del ajuste. Vea Ajustes y tolerancias.

Chaflanes y ranuras en V

Las anchuras, profundidades y ángulos de los chaflanes, y las anchuras, profundidades y ángulos de las ranuras en V, toman la clase f de ISO 2768-1 salvo que usted indique su propia tolerancia.

Qué estima el gemelo

El gemelo ejecuta los mismos toques en su pieza simulada. Su estimación es lo que el programa mecanizó en el gemelo. Cuando el valor medido y la estimación difieren, revise la máquina y el amarre: la longitud de la herramienta, el origen de pieza, el utillaje, una herramienta desgastada.

La estimación del gemelo es un cálculo hecho a partir de sus modelos físicos. No es la incertidumbre de medida de la máquina.

El informe de inspección

Cada inspección se guarda como un registro. A partir de un registro, Studio escribe el informe:

- un archivo CSV (UTF-8), para una hoja de cálculo;
- un archivo PDF, para imprimir y archivar.

El informe enumera el trabajo, el amarre, las herramientas, el palpador y su calibración y, para cada elemento: la medida nominal, la tolerancia y en qué se basa, la estimación del gemelo, el valor medido, la desviación y el veredicto. Dice por qué no se midió un elemento y muestra la pieza del gemelo vista desde arriba.

Para escribirlo, pida al asistente, por ejemplo: "Escriba el informe de la última inspección". Un mismo registro da los mismos archivos, byte por byte.

Dónde están los informes

Los informes de inspección y las hojas de amarre están en la carpeta de informes:

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

Si movió la carpeta de programas en los ajustes, la carpeta de informes está dentro de ella.

Comprobar una copia con el registro de auditoría

Studio anota en su registro de auditoría el SHA-256 de cada registro, informe y hoja de amarre. Cada entrada está encadenada a la anterior. Puede comprobar una copia que alguien le envíe:

- Pida al asistente: "Compruebe este archivo con el registro de auditoría", con la ruta del archivo.
- O ejecute, en el equipo donde se escribió:

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

La respuesta indica si el registro contiene el SHA-256 del archivo, cuándo se escribió y si la cadena del registro está íntegra. Una copia con un solo bit cambiado no se encuentra.

Hojas de amarre

Una hoja de amarre es una página para el operario en la máquina. Indica qué preparar y en qué orden. Studio escribe una para cada trabajo, en el idioma que usted usa. El papel es A4, o Letter en Estados Unidos y Canadá.

Para escribir una, pida al asistente de IA, por ejemplo: "Escriba la hoja de amarre de bracket.nc". El asistente usa la herramienta `setup_sheet`.

> **Nota**
>
> Las hojas de amarre están en la edición Pro.

Qué muestra una hoja de amarre

- Programas: El nombre, la versión y el número de líneas de cada programa, y los primeros 8 dígitos hexadecimales de su SHA-256. Compárelos con el programa cargado en la máquina.
- Bruto y sujeción: El tamaño y el material del bruto, y cómo lo monta el gemelo. Para una mordaza: los calzos paralelos, el agarre y cuánto sobresale el bruto por encima de las garras.
- Herramientas: El número, el tipo, el diámetro, los filos, el voladizo y la especificación de cada herramienta, y las operaciones que la usan.
- Cada amarre: Cómo se pone la pieza en él (dándole la vuelta, sujetándola de otra manera), dónde está el origen de pieza y los ciclos de palpado que lo encuentran. Los ciclos tienen los nombres que tienen en el panel del palpador.
- Recorrido: Los valores más bajo y más alto de X, Y y Z a los que se mueve el programa, en coordenadas de pieza.
- Espacio libre en Z: Vea Espacio libre en Z.
- Operaciones: La estrategia de cada operación y cómo entra en el material, su herramienta, velocidad, avance y profundidad, y el tiempo que tarda.
- Vista superior: El bruto, las garras de la mordaza, el origen de pieza y la silueta de cada elemento.
- Notas: Las notas del CAM, en inglés.

Los tiempos se estiman a partir del programa: sus avances, y la velocidad de avance rápido y la aceleración de la máquina. No se cuentan los cambios de herramienta. Una ejecución de prueba en el gemelo da tiempos que cuentan la aceleración y la deceleración.

Espacio libre en Z

Para cada amarre, la hoja comprueba que el palpador y cada herramienta pasan por encima del amarre. También comprueba que cada herramienta llega a los puntos más alto y más bajo del programa. Las alturas son las del presupuesto de Z del gemelo, medidas desde la superficie de la mesa.

Concepto	Tiene que alcanzar
Palpador	1 mm sobre el punto más alto del amarre, con Z cerca del extremo superior de su recorrido
Cada herramienta, arriba	0.5 mm sobre el punto más alto del amarre, y el punto más alto del programa
Cada herramienta, abajo	El punto más bajo de sus propias operaciones

Cada fila dice **Suficiente** o, en rojo, cuántos milímetros le faltan.

> **Precaución**
>
> Una fila en rojo significa que el programa no puede ejecutarse en este amarre. Use una herramienta con menos voladizo o un palpador más corto, o sujete la pieza más abajo, por ejemplo con bridas.

Una brida puede sobresalir por encima del punto más alto del programa. La hoja lo indica entonces: el programa no debe cruzar la brida. Una ejecución de prueba en el gemelo lo comprueba.

Impresión y firma

1. Abra la hoja de amarre en un navegador web.
2. Imprímala. Resultado: cada amarre después del primero empieza en una página nueva.
3. En la máquina, rellene el recuadro del final: el operario, la fecha y la inspección de la primera pieza (conforme o no conforme, y el número de su informe de inspección).

Una hoja de amarre nueva del mismo trabajo sustituye al archivo. Su SHA-256 se anota en el registro de auditoría.

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Conectar una máquina real

Borrador (ronda 14): el texto se está redactando; las secciones aún pueden crecer.

Dos maneras de trabajar con una máquina

- Controle el MC-1 desde un programa de envío propio (bCNC, gSender o Universal G-code Sender) por su puerto USB. Véase Programas de envío.
- Ponga el gemelo a su lado en modo sombra (vista previa): su programa de envío se conecta a Studio en lugar de a la máquina, Studio pasa cada línea a la máquina y ejecuta las mismas líneas en el gemelo, y compara ambos.

Enviar un programa a una máquina real desde el propio Studio no está en la 1.0.

Modo sombra (vista previa)

> **Nota**
>
> En la 1.0 el modo sombra es una vista previa que se inicia desde la línea de comandos. Se ha probado con un segundo gemelo que hace las veces de la máquina, todavía no con una real. La ventana aún no tiene un lugar donde elegir el puerto, conectar o ver la comparación; eso llegará en una versión posterior.

Iniciar el modo sombra

Necesita: el MC-1 encendido y conectado a este equipo por USB; su programa de envío instalado.

1. Busque el puerto de la máquina. En Windows: **Administrador de dispositivos > Puertos (COM y LPT)**, por ejemplo COM5.
2. Inicie Studio con ese puerto desde un símbolo del sistema:

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

   Resultado: Studio abre el puerto y el gemelo sigue a la máquina.

3. En su programa de envío, conéctese al puerto de envío de Studio, `localhost:2323`, en lugar de a la máquina. Resultado: las líneas de su programa de envío llegan a la máquina a través de Studio, y el gemelo ejecuta las mismas líneas.

El puerto de envío de Studio escucha solo en este equipo. Para acceder a él desde otro equipo, añada `--bind any`.

Qué compara el gemelo

- Las respuestas, línea por línea: cada `ok` o error, y las alarmas en su orden.
- La trayectoria: la posición que informa la máquina frente a la trayectoria del gemelo, y cuánto se adelanta o se retrasa el gemelo.
- Los ajustes: una vez, cuando la máquina está conectada y antes de que su programa de envío se conecte. Conecte su programa de envío después de que Studio se haya enlazado con la máquina.
- El controlador: una vez, junto con los ajustes. Si ambos ejecutan el mismo firmware y, con el núcleo propio de Twinwright en ambos, su versión y lo que tiene la placa de la máquina: husillo, encoder, volante, ejes, palpador, conexión. El gemelo no modela los puertos auxiliares ni la tarjeta SD de la placa: la comparación los enumera y no cuentan como una diferencia.
- Los tiempos de ejecución de las partes del programa.

La protección

Inicie Studio también con `--shadow-guard` y el gemelo comprueba cada línea de su programa de envío antes de que llegue a la máquina. Cuando el gemelo prevé un choque, retiene la línea y pone la máquina en retención de avance. La ventana muestra el estado de la protección en **Diagnóstico**.

Qué no puede llegar a la máquina

En el modo sombra solo llegan a la máquina las líneas de su programa de envío, junto con las propias consultas del gemelo, que no cambian nada (los ajustes y el estado).

- Los controles de la ventana actúan solo sobre el gemelo, y los que moverían o pondrían en marcha la máquina se rechazan mientras el gemelo sigue a una máquina. El panel del palpador está bloqueado.
- Un cliente de IA solo puede leer: ve el estado de la máquina y no cambia nada.
- Un volante externo conectado a Studio actúa solo sobre el gemelo.

> **⚠ Advertencia**
>
> En el modo sombra, la parada y la parada de emergencia de la ventana no detienen la máquina. Para detener la máquina, use su propia parada de emergencia o la parada de su programa de envío.

Programas de envío

Los programas de envío probados con el núcleo propio de Twinwright y cómo configurarlos: Programas de envío.

Cuando algo difiere

Una diferencia significa que el perfil o los ajustes del gemelo no son los de esta máquina, o que la máquina hace algo que el modelo no hace. Compruebe primero los ajustes: la comparación enumera los que difieren.

El asistente de IA

Una IA puede trabajar en el gemelo con las mismas herramientas que usted: prepara, escribe y comprueba programas y los ejecuta en la simulación. Puede hablar con una en el asistente propio de Studio o conectar un cliente de IA como Claude Desktop. Todo lo que modificaría la máquina de la ventana espera su sí. La IA no forma parte de ninguna función de seguridad.

Activar la IA

La IA está desactivada en una instalación nueva. Para activarla, haga una de estas cosas:

- haga clic en el estado de la IA en la barra superior (**IA desactivada**) y elija **Interruptor de IA de la ventana**;
- **Herramientas > Ajustes... > IA y MCP**: **interruptor de IA** y, a continuación, **Aceptar**;
- envíe un mensaje en el asistente: esto activa el interruptor.

Studio conserva el interruptor tal como usted lo deje para el próximo inicio. El menú **IA** (**Asistente de IA...**, **Actividad de IA...**, **Conectar clientes de IA...**) aparece desde el nivel Operario.

Con el interruptor desactivado, los clientes de IA pueden seguir usando máquinas de práctica en segundo plano, pero no la máquina de su ventana.

El servicio de IA y su clave

El asistente se comunica con un servicio de IA que usted elige, con su propia cuenta:

- **Anthropic (Claude)**, **OpenAI**, **DeepSeek**, **Qwen (Tongyi)**, **Kimi (Moonshot AI)**, **Zhipu (GLM)**;
- **LM Studio (local)** u **Ollama (local)**: un modelo en su propio equipo, sin necesidad de clave;
- **Personalizado (compatible con OpenAI)**: cualquier servicio con una interfaz compatible con OpenAI, en la dirección que usted escriba.

Para configurarlo:

1. Abra **IA > Asistente de IA...**
2. Elija el **Proveedor**. Studio rellena su **URL base**.
3. Pegue su clave en **Clave de API** y haga clic en **Guardar clave**. Resultado: la clave se guarda en este equipo, cifrada. En Windows solo su cuenta de Windows puede leerla. En Linux va al llavero del escritorio (Secret Service: GNOME Keyring, KWallet); sin llavero, se cifra con una clave de un archivo que solo su cuenta de usuario puede leer, y la línea bajo la clave lo indica.
4. Haga clic en **Listar modelos** y elija un **Modelo**.

Studio no escribe la clave en sus ajustes, en el Registro de Windows ni en el registro de auditoría. Para borrar una clave guardada, vacíe el campo y haga clic en **Guardar clave**.

> **Nota**
>
> Lo que usted envía al servicio de IA va a ese servicio bajo su cuenta y sus condiciones: sus mensajes y, con **Adjuntar el programa, el amarre y los resultados de comprobación de la ventana** activado, el programa, el amarre y los resultados de comprobación, y los resultados de las herramientas que usa la IA. Para mantenerlo todo en su equipo, use un modelo local.

Antes de su primer mensaje a un servicio en la nube, el asistente dice una vez qué se envía allí. Haga clic en **Enviar** para continuar o en **Usar un modelo local...** para cambiar a LM Studio; su mensaje se queda en el cuadro. **Herramientas > Ajustes... > IA y MCP** enumera los servicios que usted aceptó; **Volver a preguntar** hace que el asistente vuelva a preguntar.

El panel del asistente

IA > Asistente de IA... abre el asistente en una ventana propia.

- Escriba en el cuadro de la parte inferior. Intro envía; Mayús+Intro empieza una línea nueva. **Parar** detiene la respuesta.
- **Adjuntar el programa, el amarre y los resultados de comprobación de la ventana (si han cambiado)** da a la IA lo que hay en su ventana con cada mensaje. Está activado de forma predeterminada.
- La conversación muestra cada herramienta que usa la IA, con ✓ cuando funcionó y ✗ cuando no.
- Las imágenes que crea la IA (la pieza, la trayectoria de herramienta, la desviación respecto a la pieza, una comparación) aparecen en la conversación. Haga doble clic en una para ampliarla o guardarla.
- **Nueva conversación** empieza de cero. Las conversaciones no se guardan al cerrar Studio.

Cerrar la ventana solo la oculta; la conversación permanece hasta que usted salga.

Qué puede y qué no puede hacer la IA

Con el interruptor activado, la IA puede:

- leer el estado de las máquinas, y hacer la referencia, desbloquear, hacer desplazamientos manuales (jog) y enviar líneas de código G;
- preparar el bruto, las herramientas, el origen de pieza y la pieza, palpar e inspeccionar;
- comprobar, simular, optimizar, generar, editar y cargar programas, e iniciar o reiniciar una ejecución;
- leer ejecuciones y sus resultados, compararlos y crear imágenes de ellos;

- leer el registro de auditoría y el estado de la licencia;
- mostrarle algo en la ventana mientras le explica: un espacio de trabajo, la orientación de la vista 3D, el rastro, la trayectoria del programa, la coloración, una pestaña del panel inferior, un paso de **Preparación** (solo cambia lo que muestra la ventana);
- buscar en este manual y responder a partir de él;
- cambiar el idioma, la región, las unidades, la moneda y la tarifa por hora de máquina, solo después de que usted haga clic en **Permitir** (vea más abajo), con el mismo efecto que en **Herramientas > Ajustes...**.

Las herramientas que pertenecen a una edición de pago (inspección, informes, anidado, el optimizador) funcionan solo en esa edición.

La IA no puede cambiar los demás ajustes de Studio: el nivel de usuario, el interruptor de IA, la forma en que Studio le pregunta, la privacidad, el registro ni la apariencia; tampoco la licencia ni el registro de auditoría.

Preguntar antes de modificar la máquina

Antes de que la IA modifique la máquina de la ventana (en el menú de la IA de la barra superior y en **Herramientas > Ajustes... > IA y MCP**) tiene dos opciones:

- **Preguntarme cada vez** (la opción predeterminada);
- **No preguntar (la IA maneja la ventana directamente)**.

Cuando la IA quiere modificar la máquina de la ventana (moverla, encenderla, soltar la parada de emergencia, cargar o ejecutar un programa, cambiar el amarre o las herramientas) o uno de los ajustes que puede cambiar, se abre una barra sobre la barra de la máquina. Indica qué IA lo pide y qué quiere hacer, y hace una cuenta atrás.

1. Lea la pregunta.
2. Haga clic en **Permitir** o **Rechazar**. Con Intro no se responde: use el ratón, o pase con Tab hasta un botón y pulse la barra espaciadora.

Si no hay respuesta en 60 s, cuenta como **Rechazar**. Un sí cubre también las solicitudes posteriores de esa misma acción durante dos minutos.

Studio nunca pregunta antes de que la IA pare la máquina, retenga el avance o la reinicie, pulse la parada de emergencia o solo lea. Sobre las máquinas de práctica y las copias ocultas en las que la IA simula no se pregunta.

Solo lectura en una máquina real

Cuando el gemelo de la ventana sigue a una máquina real (modo sombra, una versión preliminar en 1.0; vea Una máquina real), la IA solo lee y analiza. No puede mover la máquina, cambiar su amarre ni sus decalajes de origen, ni ejecutar programas allí, y no pregunta: Studio rechaza esas solicitudes. Las máquinas de práctica funcionan como antes.

Actividad de IA

IA > Actividad de IA... muestra lo que hacen los clientes de IA en este equipo:

- **Máquinas en segundo plano**: las máquinas de práctica que usa la IA, que usted no ve en la ventana ("ventana" es la máquina que está en su ventana);
- **Ejecuciones**: las últimas 16 simulaciones y optimizaciones;
- **Clientes de IA conectados**: quién está conectado, cómo, y su última llamada.

Desde aquí puede detener una ejecución, ver la imagen de una ejecución, cerrar una máquina en segundo plano y detener o iniciar el host del gemelo. El host del gemelo es el programa en segundo plano que contiene las máquinas de práctica. Se inicia cuando se conecta un cliente de IA y se cierra tras 30 minutos sin uso.

La pestaña **Actividad de IA** del panel inferior enumera las llamadas de la IA a la máquina de la ventana: hora, llamada, máquina, quién y resultado.

Conectar otros clientes de IA (MCP)

Los clientes de IA que hablan MCP (Model Context Protocol) pueden usar las herramientas de Studio.

1. Abra **IA > Conectar clientes de IA...**
2. Haga clic en su cliente: **Claude Desktop**, **ChatGPT de escritorio (modos Codex y Work)**, **Cursor**, **VS Code (GitHub Copilot)** o **LM Studio (modelos abiertos locales)**. Resultado: para Claude Desktop y ChatGPT de escritorio, Studio muestra el cambio que hará en los ajustes del cliente y lo hace cuando usted hace clic en **Aceptar**; conserva al lado el archivo anterior del cliente. Los demás abren su propia página de instalación.
3. Cierre el cliente por completo y ábralo de nuevo.
4. Active el **Interruptor de IA de la ventana** en Studio.

Para cualquier otro cliente, **Otros clientes: copiar la configuración** copia los ajustes para pegarlos en él. El cliente inicia `TwinwrightStudio.exe --mcp` en segundo plano; **El comando y la configuración** muestra el comando exacto.

Clientes que solo aceptan una dirección

Para los clientes que solo aceptan una dirección, active **URL local (HTTP, para clientes que solo aceptan una URL)** en el menú de la IA y use **Copiar la URL local (HTTP)**. La dirección funciona solo en este equipo y necesita el token que la acompaña. **Token nuevo** hace que el anterior deje de funcionar.

Las herramientas, sus argumentos y sus resultados: Herramientas MCP.

Qué se registra

Herramientas > Registro de auditoría... (Ctrl+L) registra lo que hace la IA, con el nombre de la IA en la columna **Quién**. Filtre **Quién** por **IA** para ver solo las entradas de la IA; haga doble clic en una llamada para ver juntas todas sus entradas.

En el nivel predeterminado, **Estándar**, el registro conserva:

- la activación y desactivación del interruptor de IA, y los cambios en la forma en que Studio pregunta;
- cada pregunta, su respuesta y cada acción de la IA en la máquina de la ventana;
- cada llamada a una herramienta, con una forma abreviada de sus argumentos;
- las conexiones y desconexiones de clientes;
- los mensajes del asistente: los primeros 180 bytes en el resumen, hasta 2000 bytes en la entrada.

Las claves, los tokens y las contraseñas se escriben como `***`.

Idioma y unidades

Studio sigue el idioma de su sistema en el primer inicio. Puede cambiar en cualquier momento el idioma, las unidades y el aspecto de los números, las fechas y el dinero. Los programas y los archivos no cambian.

Cambiar el idioma

Elija **Herramientas > Language / 语言** y un idioma: English, 简体中文 (chino simplificado), Français, Deutsch, 日本語 o Español. El nombre del menú es el mismo en todos los idiomas, de modo que puede encontrarlo en un idioma que no lee.

La ventana cambia de inmediato; no hace falta reiniciar Studio. Las líneas que ya están en la consola permanecen en el idioma en que se escribieron.

Herramientas > Ajustes... > General > Idioma tiene una opción más, **Según Windows**: Studio sigue entonces el idioma del sistema en cada inicio. Un idioma para el que Studio no tiene paquete se muestra en inglés.

Unidades: milímetros y pulgadas

Herramientas > Unidades permite elegir **Según la región**, **Métrico (mm)** o **Pulgadas**. El cambio se aplica de inmediato.

En pulgadas, Studio muestra:

Magnitud	Métrico	Pulgadas
Longitud	mm	in
Avance	mm/min	in/min
Velocidad de corte	m/min	ft/min
Fuerza	N	lbf
Par	N·m	lbf·in
Potencia	W	hp
Temperatura	°C	°F

La velocidad del husillo sigue en rpm. Los pasos del desplazamiento manual (jog) pasan a ser 0.0005, 0.005, 0.05, 0.5 y 2 in, y las velocidades del desplazamiento manual, 4, 12, 60 y 120 in/min.

Las unidades cambian lo que Studio muestra y lee, no sus archivos. Un programa se ejecuta tal como está escrito, en `G20` o `G21`. Los programas que crea Studio se escriben en milímetros (`G21`). Las herramientas de la IA y los registros trabajan en milímetros.

Región: números, fechas y dinero

Herramientas > Ajustes... > General > Región define el aspecto de los números, el dinero y las fechas: **Según Windows (Configuración > Hora e idioma)**, **Según el idioma** o una región de la lista. En la misma página:

- **Moneda**: un código ISO 4217 como USD, EUR o CNY; vacío significa el de la región;
- **Símbolo de moneda**: vacío significa el propio de la moneda;
- **Tarifa por hora de máquina**: su coste de una hora de tiempo de máquina, para las estimaciones de coste; 0 no muestra ninguna.

La región decide el separador decimal y el separador de millares en los números de Studio, las fechas en las ejecuciones, las versiones y los informes, y el tamaño de papel de los informes (Letter en Estados Unidos y Canadá, A4 en el resto). Los números que pertenecen a la máquina, como las posiciones y los números de un programa, usan siempre un punto y ningún separador de millares.

Escribir números

Studio lee de forma estricta un número escrito en los campos donde un error costaría una pieza: los campos del CAM, las medidas y las posiciones en **Preparación**, las longitudes del palpador y la velocidad del husillo.

- Use un solo separador decimal: un punto o una coma (`13.3` o `13,3`).
- No escriba separador de millares.
- Puede escribir la unidad del campo después del número: `13 mm`, `0.5 in`, `30°`.
- También se leen los dígitos escritos con un método de entrada (dígitos de ancho completo).

`1,000` plantea una duda cuando su región escribe el separador decimal como un punto: ¿se quiso decir uno o mil? Studio no adivina; le pide que escriba el número que quiere decir. Donde la región escribe una coma, `1,000` es uno.

Qué significa “se lee como”

Cuando lo que usted escribió no está escrito como Studio escribe los números, el campo muestra lo que Studio leyó: escriba `13,3` y muestra **se lee como 13.3**. En pulgadas muestra cuatro decimales: **se lee como 0.3750**. Compruébelo antes de continuar.

Cuando Studio no puede leer lo que usted escribió, el campo se pone en rojo y una línea debajo dice por qué, por ejemplo “‘1,2,3’ no es una longitud”. Studio no toma el valor: el campo conserva el último que tomó hasta que usted lo corrija.

Fracciones de pulgada

En pulgadas, los campos de longitud también aceptan fracciones:

- `3/8`, `5/4`
- `1 1/2` o `1-1/2`
- `2 3/8"`, `1/2 in` (la marca de pulgada puede ser `"`, `”` o `in`)

El campo muestra el decimal y la fracción que leyó: **se lee como 0.3750 (3/8")**. Studio no acepta un punto decimal con una fracción (`1.5/2`), una fracción impropia después de un número entero (`1 5/4`) ni espacios alrededor de la barra.

Studio muestra también las medidas en pulgadas como fracciones, hasta 1/64 in: los diámetros y las longitudes de las herramientas, las medidas del bruto, y los diámetros y las profundidades de los agujeros se leen `1/4"` o `1 1/2"`. Una medida que no es un 1/64 muestra cuatro decimales, `0.2010 in`. Las coordenadas, los avances, las velocidades y las mediciones son siempre decimales.

En milímetros no se lee una fracción: “Los campos en milímetros no aceptan fracciones: escriba 4.7625 o cambie a pulgadas”.

Las descripciones de herramientas leen medidas en pulgadas en cualquiera de los dos sistemas, por ejemplo `flat 1/4in n2`.

Velocidad del husillo

Puede escribir la velocidad del husillo en lugar de cambiarla en pasos de 1000 rpm:

1. En **Ejecución**, abra **Manual** en la columna derecha.
2. Haga doble clic en la lectura de **Velocidad del husillo** o pulse Intro sobre ella.
3. Escriba la velocidad y pulse Intro. Con Esc, o al hacer clic en otro sitio, se conserva la velocidad anterior.

Cuando el husillo gira en manual (sin ningún programa en ejecución), la nueva velocidad se envía al husillo de inmediato. Cuando está parado, Studio conserva la velocidad para el próximo uso de **Husillo en sentido horario** o **Husillo en sentido antihorario**. Mientras se ejecuta un programa, el programa fija la velocidad y el campo no se abre.

Una velocidad fuera del rango del husillo pone el campo en rojo y no se toma. El rango del MC-1 es de 1000–24000 rpm.

Qué permanece en inglés

- el código G y los archivos que Studio escribe y lee (trabajos, amarres, grabaciones, archivos de licencia);
- las herramientas de la IA y lo que va hacia los clientes de IA y viene de ellos;
- los capítulos de referencia generados a partir del propio programa: Herramientas MCP, Línea de comandos y Esquema del perfil de máquina;
- el mensaje de cada entrada de auditoría sobre programas (el registro lo guarda también en su idioma);
- en alemán, francés, japonés y español en 1.0: las descripciones de errores y alarmas del controlador, la ayuda de los ajustes del controlador y algunos mensajes de licencia.

El PDF de un informe de inspección está en su idioma cuando sus fuentes tienen las letras de su idioma, y en inglés en caso contrario; su archivo CSV está siempre en su idioma.

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Teclado y accesibilidad

La máquina, los menús, los botones de los paneles y el editor funcionan con el teclado. Girar y hacer zoom en la vista 3D, elegir un punto en ella y cambiar el tamaño de los paneles requieren un ratón. En Windows, un lector de pantalla lee los botones y las lecturas de los paneles.

Ayuda > Atajos de teclado (o **?**) enumera todas las teclas de Studio.

Las teclas de la máquina

Tecla	Acción
F1	**Parada**: pulse; vuelva a pulsar para soltarla
F2	**Encender**
F3	**Referencia**
F4	**Desbloquear**
F5	**Inicio de ciclo**
F6 o Espacio	**Retención de avance**
F7	**Parar**: retención de avance y, cuando la máquina se detiene, un reset; vuelva a pulsar para hacer el reset de inmediato
F8	Activar o desactivar **Refrigeración**
F9	**Husillo en sentido horario**
F10	**Husillo en sentido antihorario**
F11	**Parar husillo**
F12 o Ctrl+X	**Reset**

Desplazamiento manual (jog) y volante:

Tecla	Acción
← →	Desplazamiento manual en X
↓ ↑	Desplazamiento manual en Y
AvPág, RePág	Desplazamiento manual en Z
Mayús con cualquiera de ellas	Desplazamiento manual mientras mantiene pulsada la tecla
1–5	Paso de 0.01, 0.1, 1, 10 o 50 mm
6	**Jog continuo**: cada tecla de desplazamiento manual mueve la máquina mientras la mantiene pulsada
Ctrl+1 a Ctrl+4	Velocidad de desplazamiento manual de 100, 300, 1500 o 3000 mm/min
Esc	**Parar desplazamiento manual**
X, Y, Z	Elegir el eje del volante
, y .	Girar el volante una muesca hacia abajo o hacia arriba (con Mayús: diez)

Porcentajes (override):

Tecla	Acción
Ctrl+↑, Ctrl+↓	Avance +10 %, −10 %
Ctrl+→, Ctrl+←	Husillo +10 %, −10 %
Ctrl+0	Avance y husillo de nuevo al 100 %

Cada pulsación de una tecla de desplazamiento mueve un paso. Un desplazamiento mantenido se detiene cuando suelta la tecla o cuando la ventana pierde el foco. En pulgadas, los pasos son de 0.0005–2 in y las velocidades de 4–120 in/min.

Las teclas F funcionan en cualquier parte. Las demás teclas de la máquina no funcionan mientras escribe en un cuadro de texto, una lista o el editor: haga clic en la vista 3D para devolvérselas a la máquina.

Las teclas de dígitos y las teclas , y . funcionan según su posición en el teclado, de modo que sirven con cualquier distribución de teclado.

Parada de emergencia

Pulse F1 o haga clic en el botón rojo redondo del extremo derecho de la barra de la máquina. En el gemelo, esto abre el circuito de parada de emergencia como lo hace el botón de la máquina: los accionamientos y el variador del husillo pierden la alimentación, y el controlador genera AL ARM:10. Vuelva a pulsar F1 para soltarla y siga los pasos para volver a trabajar: **Encender**, **Reset**, **Referencia**.

> ⚠ **Advertencia**
>
> F1 y el botón rojo detienen el gemelo. No son una parada de emergencia para una máquina real. En una máquina real, use el botón de parada de emergencia de la propia máquina.

Las teclas de la ventana

Tecla	Acción
Alt+1 a Alt+6	Ir a **Preparación**, **CAM**, **Programa**, **Ejecución**, **Análisis** o **Diagnóstico**
Ctrl+Tab, Ctrl+Mayús+Tab	El espacio de trabajo siguiente o el anterior
Ctrl+J	Plegar o desplegar el panel inferior
Ctrl+Mayús+M	Cambiar de máquina
Ctrl+O	**Abrir un programa…**
Ctrl+P	**Imprimir programa…**
Ctrl+,	**Ajustes…**
Ctrl+L	**Registro de auditoría…**
Ctrl+Q	**Salir**
? o /	**Atajos de teclado**
Alt y la letra de un menú	Abrir ese menú; Alt solo resalta los títulos de los menús
Tab, Mayús+Tab	Moverse entre los botones de un panel; Espacio pulsa el botón que tiene el marco
Esc	Cerrar una ventana emergente o una subventana

> **Nota**
>
> F1 es la parada de emergencia, no la ayuda. Para abrir el manual, use **Ayuda > Documentación** o el botón **?** de un panel.

El editor del espacio de trabajo **Programa** tiene sus propias teclas:

Tecla	Acción
Ctrl+E	Ir al editor y volver a la máquina
Ctrl+K	**Cargar y comprobar**
Ctrl+S	**Guardar versión**
Ctrl+Mayús+S	Guardar el programa como archivo
Ctrl+D	**Comparar** con una versión guardada
Ctrl+F, Ctrl+G, Ctrl+H	Buscar, ir a una línea, reemplazar
Ctrl+B	**Punto de interrupción** en la línea
Ctrl+Mayús+B	**Pausar si** se cumple una condición
Ctrl+.	Aplicar la sugerencia de la comprobación para la línea
Alt+↓, Alt+↑	El hallazgo o la diferencia siguiente o anterior
Mayús+F10 o la tecla de menú	El menú de la línea
Ctrl+Mayús+O	**Optimizar**
Ctrl+Mayús+G	**Generar…**
Ctrl+Mayús+R	Tomar el programa que está en la máquina
Esc	Cerrar la comparación

La ventana del manual: Ctrl+F busca, Alt+← y Alt+→ van hacia atrás y hacia adelante, Esc la cierra.

Mientras coloca una pieza en **Preparación**: las teclas de flecha la mueven (con Mayús, pasos más finos; con Ctrl, más gruesos), Q y E la giran, Intro la coloca, Esc la devuelve a donde estaba.

La lista completa

Ayuda > Atajos de teclado muestra todas las teclas anteriores en una sola lista. Estos comandos no tienen tecla; use sus menús:

- **Bloque a bloque** y **Bloque siguiente** (el menú **Programa** y la barra de la máquina);
- **Comprobar el programa**;

- poner a cero los ejes e ir al cero (**Máquina > Origen de pieza**);
- los movimientos del sensor de longitud de herramienta y del palpador (**Máquina > Sensor de longitud de herramienta y palpador**);
- la velocidad del husillo en pasos de 1000 rpm y el override de los movimientos rápidos (**Máquina**);
- poner en pausa la simulación y su velocidad (**Simulación**).

Usar un lector de pantalla

En Windows, Studio funciona con lectores de pantalla a través de la interfaz de accesibilidad del sistema. En cada uno de los paneles de Studio, el lector recibe:

- el nombre de cada botón, de qué tipo de botón se trata, si está activado, abierto o no disponible, su atajo y su descripción emergente;
- las lecturas como texto: la barra de la máquina (estado, programa, progreso, el siguiente paso, el reloj), la barra superior (máquina, estado, IA, nivel, edición) y la columna derecha (posición, husillo, indicadores, la línea modal);
- cada movimiento del foco con Tab;
- los menús: el texto de cada elemento, su atajo, si no está disponible o está marcado, y en cuál se encuentra.

Dos cosas se anuncian en el momento en que ocurren, esté donde esté el foco: una alarma (con el siguiente paso) y la solicitud de una IA para modificar la máquina.

La ventana del manual proporciona al lector el texto de la página, con sus encabezados y enlaces.

En 1.0:

- los demás cambios del estado de la máquina no se anuncian: lleve el foco a la barra de la máquina para oírlos;
- el listado, la consola, los mensajes, el editor y la tabla de herramientas los lee el lector según sus propias reglas; las curvas no se leen;
- en Linux no se admiten lectores de pantalla.

Qué lee en la vista 3D

La vista 3D en sí no da nada al lector. Los números que muestra están en la columna derecha y en la barra de la máquina, que el lector sí lee. Cuando usted hace clic en un punto de la pieza, se lee la frase situada bajo la vista (la línea del programa y la posición).

Texto grande y modo taller

Studio sigue la escala de pantalla que usted establece en Windows, pantalla por pantalla. No tiene un ajuste propio de tamaño de texto.

Ver > Apariencia > Modo taller (grande) agranda la barra de la máquina, sus botones, la parada de emergencia, el texto, el listado y la posición, y da al espacio de trabajo **Ejecución** una disposición para leer a distancia de brazo (vea Claro, oscuro y modo taller).

Cuando las animaciones del sistema están desactivadas, la vista 3D salta a una vista nueva en lugar de deslizarse hasta ella.

Solución de problemas

Qué hacer cuando algo no funciona como usted espera. Cada sección dice qué ve, qué suele significar y dónde está la respuesta completa. Si ninguna encaja, recopile la información de diagnóstico y escríbanos (la última sección).

Studio no se inicia

- Studio necesita Windows 10 u 11 de 64 bits; el instalador se detiene en otras versiones. Vea Requisitos del sistema.
- Si la ventana se abre pero la vista 3D queda vacía o indica que no está disponible: en **Herramientas > Ajustes... > Archivos**, marque **Renderizado por software (si la tarjeta gráfica da problemas; en el próximo inicio)** y vuelva a iniciar Studio. Actualizar el controlador de la tarjeta gráfica suele solucionar la causa. Vea Cuando la vista 3D no puede iniciarse.
- Si Studio se cerró mientras se iniciaba, el siguiente inicio ofrece el informe (vea Studio se cerró inesperadamente).

La máquina está en alarma

En una alarma, el estado de la barra de la máquina se pone en rojo. La barra nombra la alarma y lo que significa, y da el **Siguiente paso** con el botón que lo realiza; **Todos los pasos** enumera el camino de vuelta al trabajo cuando hay más de uno. Haga clic en el botón o use **Máquina > Realizar el paso siguiente**: Studio vuelve a calcular los pasos después de cada uno.

- Un movimiento de palpado que no tocó nada se detiene con ALARM:5: realice los pasos (**Reset**, **Desbloquear**) y compruebe después dónde esperaba el ciclo encontrar la cara.
- Un movimiento que sobrepasa el recorrido de la máquina se detiene en el límite por software: mire la línea que el listado marca en rojo y el origen de pieza con el que se ejecutó.
- Después de una alarma, puede que ya no se pueda confiar en la posición (una línea bajo **Posición** lo indica): haga de nuevo la referencia antes de mecanizar.

Los pasos para cada caso: Alarmas y qué hacer. Todas las alarmas, sus causas y cómo se eliminan: Alarmas.

Una línea del programa responde con un error

`error:` y un número significan que el controlador rechazó esa línea: no la ejecutó y la máquina no se movió por ella. El mensaje da el número y lo que significa.

- **Programa > Comprobar el programa** envía todo el programa por el modo de comprobación del controlador mientras la máquina no se mueve, y enumera las primeras líneas que rechazó: "línea 5: error:22 (...)". Corrija esas líneas y vuelva a comprobar.
- Las causas más comunes son un código que el controlador no acepta, una palabra que falta (un arco sin su centro o radio, un movimiento de corte antes de cualquier `F`) o dos códigos en una línea que no pueden ir juntos.

Todos los errores y sus causas: Errores.

La comprobación del programa encuentra un problema

El editor comprueba el programa mientras usted escribe, cuando se carga y después de una ejecución. Cada hallazgo aparece bajo su línea con el número de su regla y su tipo: un error, una advertencia o una nota.

- Un **error** es una línea que el controlador rechazaría o un movimiento que el amarre no admite; corríjalo antes de ejecutar.
- Una **advertencia** es una línea que se ejecuta pero puede que no haga lo que usted quiere: un movimiento rápido por debajo de la parte superior de la pieza, un husillo arrancado sin `S`, un cambio de herramienta tras el cual el programa baja antes de estar sobre su punto de inicio.
- La parte de vista previa de la comprobación necesita la máquina encendida y con la referencia hecha, y ningún programa en ejecución; hasta entonces espera.

Las reglas y cómo corregir un hallazgo: La comprobación del programa. Lo que comprueba además una ejecución en el gemelo: Qué comprueba el gemelo.

El ciclo de palpado no se inicia

Iniciar palpado... permanece en gris mientras una comprobación de la tarjeta del palpador está en rojo o mientras falta algo; la línea bajo los botones dice entonces qué (indique X e Y, indique el diámetro del anillo, espere a que termine una llamada en curso).

- **El husillo no tiene palpador**: en **Preparación**, paso **Herram.**, elija el palpador y haga clic en **Cambiar herramienta**.
- **El husillo gira**: deténgalo (**Parar husillo**, F11).
- **No hay espacio sobre el bruto para pasar por encima**: la línea indica la altura de la punta del palpador con Z en su posición más alta, la parte superior del amarre y cuánto falta. Use un palpador más corto o sujete el bruto más abajo: bridas o cinta de doble cara en lugar de la mordaza.
- **El gemelo sigue a una máquina real**: palpe allí con el propio control de la máquina.
- Un ciclo que no tocó nada se detiene con ALARM:5 (vea La máquina está en alarma).

Vea Cuando un ciclo se detiene.

Licencia y activación

Ayuda > Licencia muestra la edición vigente, el estado de la licencia y cuándo termina.

- **Activación en línea**: escriba el código tal como lo recibió; **Activar** está disponible en cuanto lo escrito se reconoce como un código. La respuesta del servidor aparece debajo. Cuando la respuesta es que la licencia está en demasiados equipos, ha terminado, está en pausa, sin pagar o reembolsada, **Abrir la página de licencias** la abre en nuestro sitio web: allí puede liberar un equipo, renovar o gestionar la suscripción.
- **Sin internet**: el mismo cuadro de diálogo tiene los pasos sin conexión. Studio proporciona un código de solicitud; en un equipo con internet usted obtiene en la página de licencias el archivo de licencia correspondiente y después importa ese archivo aquí.
- Una suscripción que no se renueva sigue funcionando 14 días más (la licencia muestra **En periodo de gracia**). Pasado ese plazo, o cuando una licencia termina, Studio vuelve a ser la edición Learning; sus proyectos y programas quedan intactos.

Vea Licencias y Cuando una licencia termina.

El asistente de IA no responde

- La línea bajo el cuadro de entrada indica qué está haciendo el asistente (esperar al servicio, el gemelo trabajando en una herramienta).
- Compruebe el **Proveedor**, la **URL base** y el **Modelo**; para un servicio en la nube, que haya una clave guardada (**Guardar clave**).
- Para un modelo local, su servidor debe estar en marcha (el servidor de LM Studio, Ollama) y debe haber un modelo elegido (**Listar modelos**).
- Una respuesta puede tardar; **Parar** la termina.
- El primer mensaje a un servicio en la nube espera su decisión: Studio muestra una vez qué se envía y solo lo envía cuando usted acepta (se pregunta una vez por servicio). **Usar un modelo local…** mantiene todo en este equipo. Vea Privacidad.
- Cuando la IA quiere modificar la máquina de la ventana, una barra en la parte inferior de la ventana se lo pregunta. Si no hay respuesta en 60 s, la solicitud cuenta como rechazada.

Vea El asistente de IA.

Studio se cerró inesperadamente

En el siguiente inicio, Studio pregunta qué hacer con el informe de lo ocurrido: abrir su carpeta o guardarlo como archivo. No se envía nada. El informe omite programas, nombres de archivos y carpetas, claves, códigos y nombres de personas; los informes de más de 30 días se eliminan al iniciar. Vea Privacidad.

Obtener ayuda

Ayuda > Informar de un problema… envía al soporte de Twinwright un problema, una pregunta sobre cómo usar algo o una idea. Diga qué hizo, qué esperaba y qué vio; indique una dirección de correo electrónico si quiere recibir allí también las respuestas. Siempre se envía con él, y se muestra en el cuadro de diálogo (**Ver todo…** lo muestra completo): la versión y la compilación, la edición, el sistema, su idioma y su pantalla, y las últimas 50 entradas del registro de auditoría resumidas —qué se hizo y cómo terminó, sin el contenido de ningún programa, con su propia carpeta escrita como `~`—. **Adjuntar el paquete de diagnóstico** y **Adjuntar una captura de la ventana** no están marcadas al abrirse el diálogo; no se envía nada de ellas a menos que usted las marque. Tras **Enviar**, el diálogo muestra el número del ticket (por ejemplo, `TW-1042`). Sin conexión, el informe se guarda como archivo, `problem-<time>.zip`, en `Documentos\Twinwright Studio\Reports`: vuelva a enviarlo desde el diálogo o súbalo en la página de soporte del sitio web. **Guardar como archivo…** lo guarda sin enviarlo.

Un cliente de IA puede redactarle un borrador de informe (report_draft): se abre el mismo diálogo con su texto y una línea que indica qué cliente lo escribió. Léalo y cambie lo que quiera; solo usted puede enviarlo, y solo usted puede adjuntar el paquete o la captura.

Ayuda > Mis informes enumera los informes enviados desde este equipo: el número de cada uno, su estado (recibido, en curso, respondido, resuelto) y cuándo cambió por última vez; un punto marca una respuesta que usted no ha leído. Elija uno para leer la conversación; **Añadir algo…** envía más al mismo ticket (**Adjuntar el paquete de diagnóstico…** le añade el paquete). Las respuestas también están en "Mis tickets" del centro de clientes. Mientras un informe no está resuelto, Studio busca respuestas nuevas cada 30 minutos y el menú Ayuda muestra cuántas son nuevas; **Ajustes > General > Buscar respuestas a mis informes cada 30 minutos** lo desactiva. Qué se envía y qué se conserva: Privacidad.

Ayuda > Recopilar información de diagnóstico… reúne lo que necesita el soporte en un único archivo ZIP en `Documentos\Twinwright Studio\Reports`: la versión, los ajustes (las claves se escriben como `***`), el archivo de la máquina, los ajustes del controlador, los últimos 7 días del registro de auditoría, los informes de fallos y el estado de la licencia. **El programa y el amarre actuales** se omite a menos que usted lo marque. Se eliminan las rutas, los nombres de archivo y el nombre de usuario del equipo.

Abra el archivo y revíselo antes de enviarlo; este archivo no se envía por sí solo (un informe lo lleva solo cuando usted marca **Adjuntar el paquete de diagnóstico**). La dirección a la que escribir está en **Ayuda > Acerca de**.

Operations manual

Requisitos del sistema

Lo que necesita un equipo para ejecutar Twinwright Studio y lo que la vista 3D necesita de los gráficos.

Windows

- Windows 10 o Windows 11 de 64 bits (x64). El instalador se detiene en otras versiones.
- No hay que instalar nada más: Studio es un único archivo de programa y solo usa las bibliotecas que vienen con Windows.

Linux

- Linux en x86-64 con glibc 2.38 o posterior: Ubuntu 24.04, Debian 13, Fedora 39 y posteriores, y las distribuciones basadas en ellas.
- Un escritorio con X11 o Wayland.
- Para el AppImage: FUSE 2 (en Ubuntu, el paquete `libfuse2t64`). Sin él, use el `.tar.gz`.
- Para la activación y las pruebas en línea: libcurl (en Ubuntu, el paquete `libcurl4t64`; los escritorios suelen tenerlo). Sin ella, Studio se activa sin conexión.
- Para las claves de los servicios de IA: el llavero del escritorio (Secret Service: GNOME Keyring o KWallet). Sin llavero, Studio guarda las claves en un archivo que solo usted puede leer.

Gráficos

La vista 3D usa Direct3D 11, en cualquier tarjeta gráfica o gráficos integrados que lo tengan (nivel de características 10.0 o superior). Sin ello, Studio dibuja con el renderizador por software de Windows (WARP): funciona, pero la vista es más lenta (como máximo 30 fotogramas por segundo) y no hay primer plano de la herramienta.

Ayuda > Acerca de muestra, en **Pantalla**, cuál de los dos usa Studio.

Cuando la vista 3D no puede iniciarse

El resto de la ventana funciona sin la vista 3D. Si la vista queda vacía o indica que no está disponible:

1. Abra **Herramientas > Ajustes… > Archivos**.
2. Marque la casilla **Renderizado por software (si la tarjeta gráfica da problemas; en el próximo inicio)**.
3. Cierre Studio y vuelva a iniciarlo.

Actualizar el controlador de la tarjeta gráfica suele resolver la causa.

Linux

En Linux, la vista 3D usa OpenGL 4.1 o posterior: los controladores Mesa lo tienen para los gráficos de los últimos diez años. Sin controlador gráfico, el renderizador por software de Mesa dibuja la vista, más despacio; para probarlo, inicie Studio desde un terminal con `LIBGL_ALWAYS_SOFTWARE=1` delante del nombre del programa.

Memoria y disco

- Memoria: 8 GB o más.
- Disco: unos 62 MB para el programa y su manual, y espacio para sus datos. Sus datos crecen con el uso: el registro de auditoría se conserva por defecto 180 días y hasta 2 GB (**Herramientas > Ajustes… > Registro**), y se conservan todas las versiones de los programas y todos los informes (consulte Sus datos y copias de seguridad).

Pantallas y escala

- Como mínimo 1280 × 720; Studio está diseñado para 1600 × 1000 o más.
- Cualquier escala de pantalla. Studio sigue la escala de cada pantalla, y al mover la ventana a otra pantalla se vuelve a escalar.

Una máquina real

En la 1.0, conectarse a una máquina real es una versión preliminar, desde la línea de comandos (consulte Una máquina real): un puerto serie (USB) a 115 200 baudios, o un controlador en la red.

Studio usa la red de este modo:

- los programas de envío de código G se conectan al gemelo de este equipo, por el puerto 2323;
- la vista en el navegador usa el puerto 8765, en este equipo;
- los clientes de IA que aceptan una dirección usan el puerto 8766, en este equipo, cuando usted lo activa;
- la activación y la renovación en línea van al servidor de licencias.

Los puertos de Studio escuchan solo en este equipo: otros equipos no pueden acceder a ellos.

En Linux, el puerto serie pertenece al grupo `dialout`: añada su cuenta una vez (`sudo usermod -aG dialout $USER`) y vuelva a iniciar sesión.

Instalar y desinstalar

Cómo instalar Twinwright Studio, sin preguntas y en muchos equipos a la vez, qué va dónde y cómo desinstalarlo.

Para Linux, consulte En Linux.

Instalar

1. Ejecute `TwinwrightStudio-<version>-setup.exe`.
2. Lea el contrato de licencia y haga clic en **I Agree**.
3. Conserve la carpeta (`%LOCALAPPDATA%\Programs\Twinwright Studio`) o elija otra.
4. En la última página, elija si desea iniciar Studio y si desea poner un acceso directo en el escritorio.

El instalador se muestra en inglés o en chino, según el idioma del sistema.

Para usted o para todos los usuarios del equipo

El instalador instala Studio para la persona que lo ejecuta. No necesita permisos de administrador, y la entrada del menú Inicio y el acceso directo del escritorio son de esa persona. Cada persona que usa el equipo instala Studio para sí misma.

La licencia pertenece al equipo, no a la persona: todos los usuarios del equipo la comparten (consulte La carpeta de licencias).

Studio no cambia ninguna asociación de archivos, ni `PATH`, ni la configuración del firewall, y no instala ningún servicio.

Instalar sin preguntas

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

- `/S` instala sin mostrar páginas y no inicia Studio.
- `/DESKTOP` añade el acceso directo del escritorio.
- `/D=` fija la carpeta. Debe ser el último argumento, sin comillas, aunque la carpeta tenga espacios.

Para desinstalar Studio sin preguntas: `uninstall.exe /S` en la carpeta del programa.

Qué va dónde

Qué	Dónde
El programa	`%LOCALAPPDATA%\Programs\Twinwright Studio`
Su trabajo: programas, versiones, informes, el registro de auditoría	`Documentos\Twinwright Studio`
El estado propio del programa: ajustes, máquinas, claves de IA	`%LOCALAPPDATA%\Twinwright\Studio`
La licencia	`%ProgramData%\Twinwright\licence`

La carpeta del programa

`TwinwrightStudio.exe`, el manual (`manual\`), `README.txt`, `THIRD-PARTY-NOTICES.txt` y `uninstall.exe`. La carpeta no cambia con las actualizaciones, de modo que los clientes de IA que usted conectó siguen funcionando.

Sus documentos

`Documentos\Twinwright Studio` (la carpeta Documentos que le da Windows, también cuando está redirigida):

- `programs\`: las versiones de cada programa y las versiones de los trabajos de CAM;
- `Reports\`: informes de inspección, hojas de amarre y paquetes de diagnóstico;
- `logs\`: el registro de auditoría.

Herramientas > Ajustes... > Archivos y **Registro** pueden mover los programas y el registro a otro lugar (en el próximo inicio).

El estado propio del programa

`%LOCALAPPDATA%\Twinwright\Studio`: los ajustes, la posición de la ventana, el mundo de cada máquina (su pieza, sus herramientas, sus decalajes y los ajustes de su controlador), las claves de los servicios de IA (cifradas para su cuenta de Windows) y los informes de fallos (se conservan 30 días).

La carpeta de licencias

`%ProgramData%\Twinwright\licence` contiene los archivos de licencia, pruebas incluidas, para todos los usuarios del equipo. Studio la crea en la primera activación o prueba. Si Studio no puede escribir allí, un administrador puede crear la carpeta y permitir que los usuarios escriban en ella.

Para configuraciones especiales, estas variables de entorno mueven las carpetas:

Variable	Mueve
`TWINWRIGHT_STATE_DIR`	El estado propio del programa

Variable	Mueve
`TWINWRIGHT_WORKDIR`	La carpeta de programas
`TWINWRIGHT_LOGDIR`	El registro de auditoría
`TWINWRIGHT_LICENCE_DIR`	La carpeta de licencias

Actualizar a una versión nueva

Cierre Studio y ejecute después el instalador de la versión nueva. Se instala en la misma carpeta; su trabajo, sus ajustes, sus máquinas y su licencia quedan como están. Studio no comprueba por sí mismo si hay actualizaciones (consulte Actualizaciones).

Si todavía hay una copia de Studio ejecutándose en segundo plano (un cliente de IA puede haber iniciado una), el instalador aparta el archivo del programa antiguo y lo elimina en la siguiente actualización o desinstalación.

Desinstalar

Use **Configuración > Aplicaciones > Twinwright Studio > Desinstalar** en Windows, o ejecute `uninstall.exe` en la carpeta del programa.

Lo que queda tras desinstalar

Al desinstalar se eliminan el programa, sus accesos directos y su entrada en **Aplicaciones**. Quedan:

- su trabajo en `Documentos\Twinwright Studio`;
- el estado propio del programa en `%LOCALAPPDATA%\Twinwright\Studio`;
- la carpeta de licencias;
- la entrada `twinwright` en los ajustes de los clientes de IA que usted conectó.

Elimine estos elementos a mano para quitarlo todo. Eliminar la carpeta de licencias no devuelve una prueba: el servidor de licencias da a cada equipo una prueba de cada edición.

En Linux

Instalar

Studio para Linux es un solo archivo, un AppImage. No se instala nada en el sistema.

1. Compruebe el archivo: `sha256sum TwinwrightStudio-<version>-x86_64.AppImage` debe dar el SHA-256 de la página de descarga.
2. Póngalo donde pueda quedarse, con un nombre sin versión, para que los clientes de IA que conecte sigan funcionando tras una actualización, y hágalo ejecutable:

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

1. Inícielo: `~/Applications/TwinwrightStudio.AppImage`, o haga doble clic en él en el gestor de archivos.

El AppImage necesita FUSE 2. Si no arranca por falta de FUSE 2, instálelo (`sudo apt install libfuse2t64` en Ubuntu 24.04 y posteriores), o use el `.tar.gz`: descomprímalo en cualquier sitio e inicie `TwinwrightStudio` en su carpeta.

Qué va dónde en Linux

Qué	Dónde
El programa, con su manual	el AppImage, donde lo haya puesto
Su trabajo: programas, versiones, informes, el registro de auditoría	`~/Documents/Twinwright Studio`
El estado propio del programa: ajustes, máquinas	`~/.local/state/twinwright/studio` (bajo `$XDG_STATE_HOME` si está definido)
Las claves de los servicios de IA	el llavero del escritorio; sin llavero, un archivo de la carpeta de estado que solo usted puede leer
La licencia	`/etc/twinwright/licence`, o `~/.local/share/Twinwright/licence`

La carpeta de licencia es del ordenador, para todos los usuarios, cuando su cuenta puede crearla o escribir en ella. Si no, Studio guarda la licencia en `~/.local/share/Twinwright/licence` (bajo `$XDG_DATA_HOME` si está definido): solo para usted, y sigue valiendo solo para este ordenador. Para compartir una licencia entre los usuarios de un ordenador, cree la carpeta una vez con `sudo install -d -m 1777 /etc/twinwright/licence`.

Prueba y activación en Linux

Las pruebas, la activación y la renovación van por internet como en Windows, con la libcurl del sistema. Sin libcurl o sin internet, active sin conexión (véase Licencias).

Actualizar y desinstalar en Linux

Para actualizar, salga de Studio y ponga el nuevo AppImage en lugar del antiguo, con el mismo nombre (paso 2 de arriba). Para desinstalar Studio, borre el AppImage. En ambos casos quedan sus documentos, la carpeta de estado, la carpeta de licencia, las claves de su llavero y la entrada `twinwright` en los clientes de IA que conectó; bórrelos a mano para quitarlo todo.

Licencias, puestos y activación

Sin licencia, Studio funciona como la edición Learning. Una licencia de pago añade las funciones de su edición. Este capítulo es para la persona que compra, activa y administra las licencias. Los precios están en el sitio web.

Si cuando lea esto las licencias de pago aún no están a la venta, la página de precios del sitio web lo indica. Hasta entonces, Studio funciona como la edición Learning y puede probar Maker o Pro gratis durante 30 días (vea Probar una edición de pago); las secciones sobre la activación se aplican cuando las licencias estén a la venta.

Ediciones

Edición	Añade	Quién puede usarla
Learning	Todo el gemelo, el editor, la máquina real, los ejemplos, la conexión con IA y el CAM para cajeras, contornos, taladrado, grabado y acabado 3D en barrido paralelo	Cualquiera, gratis, para aprender, enseñar y proyectos personales que no generen ingresos
Maker	Desbaste adaptativo, todas las estrategias de acabado 3D, mecanizado de restos, dos caras y varias fases	Una persona, incluida la venta de lo que esa persona fabrica
Pro	Todo lo de Maker, más el anidado, el optimizador de programas y la inspección con palpador (comparación con el diseño) con sus informes	Su organización, para cualquier uso comercial, por puesto

Una licencia de **Propietario de MC-1** es Pro para un puesto, para usar con la MC-1 cuyo número de serie indica. Pasa al siguiente propietario junto con la máquina.

Una licencia de pago puede ser de estos tipos:

- **Suscripción**: por meses o por años, hasta que la cancele.
- **Compra perpetua**: suya para siempre. Cubre todas las versiones publicadas en los 12 meses siguientes a la compra, o a una renovación de pago de las actualizaciones.
- **Propietario de MC-1**: como una compra perpetua, ligada a la máquina.

Puestos y equipos

Un puesto de pago lo usa una sola persona a la vez, en un máximo de tres equipos. Un equipo que comparten varias personas cuenta una sola vez.

Cada activación vincula la licencia a un equipo mediante un hash de la huella del equipo. La huella en sí nunca sale del equipo. Para ver en qué equipos está una licencia, abra la página de licencias, twinwright.com/es/licence, y escriba el código de activación.

Probar una edición de pago

Puede probar gratis cada edición de pago durante 30 días, una vez en cada equipo.

1. Use una función de pago, por ejemplo el desbaste adaptativo en el panel de CAM. Resultado: Studio indica qué edición necesita la función.
2. Elija **Probar 30 días...**.
3. Lea qué añade la prueba y hasta cuándo, y elija **Empezar la prueba**. Resultado: Studio pide la prueba al servidor de licencias de Twinwright y las funciones de la edición están disponibles hasta la fecha indicada.

Para una prueba, Studio envía la edición, un hash de la huella del equipo y la versión de Studio, nada más. El servidor da a cada equipo una prueba de cada edición: si reinstala Studio durante la prueba, al empezarla de nuevo recupera la misma fecha de fin. Al terminar la prueba, Studio vuelve a ser la edición Learning; sus proyectos y programas no cambian.

En un equipo sin internet, Studio muestra en su lugar un código de solicitud. En un equipo con internet, abra la página de licencias, twinwright.com/es/licence, elija la edición en "Probar en un equipo sin internet", pegue el código de solicitud y descargue el archivo de prueba. Después importe el archivo en el primer equipo, como en Activar un equipo sin internet.

Activar en línea

Necesita el código de activación del correo que recibió con la licencia, con la forma `TW-XXXXX-XXXXX-XXXXX-XXXXX`, y conexión a internet.

1. Elija **Ayuda > Licencia...**.
2. Elija **Activar en línea...**.
3. Escriba el código de activación. Mayúsculas, minúsculas, espacios y guiones no importan.
4. Elija **Activar**. Resultado: el archivo de licencia está en la carpeta de licencias y el cuadro de diálogo muestra la edición.

La activación solo envía el código de activación, un hash de la huella de este equipo y la versión de Studio.

Si la licencia ya está en tantos equipos como permite, libere primero uno en la página de licencias (vea Pasar una licencia a otro equipo).

Activar un equipo sin internet

Use este procedimiento para un equipo de laboratorio o de taller que nunca se conecta. Necesita el código de activación, un segundo equipo con internet y una forma de llevar un archivo pequeño, por ejemplo una memoria USB.

1. En el equipo sin internet, elija **Ayuda > Licencia...**.

2. Elija **Activar sin conexión...**.
3. Elija **Copiar**. Resultado: el código de solicitud está en el portapapeles. Contiene un hash de la huella de este equipo y la versión de Studio, ningún dato personal.
4. Pegue el código de solicitud en un archivo de texto y llévelo al equipo con internet.
5. Allí, abra twinwright.com/es/licence. En "Un equipo sin internet", pegue el código de activación y el código de solicitud.
6. Elija "Descargar el archivo de licencia". Resultado: se descarga un archivo con un nombre como `twinwright-TWL-….twlic`.
7. Lleve el archivo de vuelta. En Studio, elija **Ayuda > Licencia...** e **Importar un archivo de licencia...**, y abra el archivo. Resultado: el archivo de licencia está en la carpeta de licencias.

Un equipo sin internet no renueva la licencia por sí solo. Repita estos pasos:

- antes de que termine un periodo de suscripción;
- con una compra perpetua, una vez pasados los primeros 30 días (el periodo de reembolso), para obtener el archivo permanente;
- después de comprar más años de actualizaciones.

La carpeta de licencias

La carpeta de licencias pertenece al equipo, no a una cuenta de Windows. Una licencia que esté en ella vale para todas las personas que usan el equipo.

Sistema	Carpeta
Windows	`%ProgramData%\Twinwright\licence`
Linux	`/etc/twinwright/licence`; donde no se puede escribir: `~/.local/share/Twinwright/licence`

La variable de entorno `TWINWRIGHT_LICENCE_DIR` indica otra carpeta. Para abrir la carpeta, elija **Ayuda > Licencia...** y **Abrir la carpeta de licencias**.

La carpeta contiene los archivos de licencia (`*.twlic`, también los de las pruebas) y un archivo con la última hora en que Studio la comprobó. Puede contener varios archivos de licencia, por ejemplo una licencia de propietario de MC-1 y una suscripción. Rige la edición más alta de todos los archivos de licencia válidos.

> **Nota**
>
> No modifique estos archivos. Un archivo de licencia modificado deja de contar: descárguelo de nuevo desde la página de licencias e impórtelo.

Si Studio no puede crear la carpeta o escribir en ella, un administrador la crea y da a los usuarios permiso de escritura. En Linux, importe como root (`sudo`) o defina `TWINWRIGHT_LICENCE_DIR`.

Renovación y los 14 días de gracia

Con **Renovar antes de que venza (solo envía el número de licencia, un hash de la huella de este equipo y la versión de Studio)** activado, Studio renueva la licencia por sí solo cuando está en línea:

- una suscripción, desde cinco días antes de que termine su periodo;
- una compra perpetua, al final del periodo de reembolso y, si ha comprado más años, desde 14 días antes de que terminen sus actualizaciones.

Studio lo intenta una vez al iniciarse, como máximo una vez al día. Un intento fallido no le interrumpe. Solo se envía lo que nombra la opción.

Si una suscripción no se renueva, todo sigue funcionando 14 días más. La licencia muestra **En periodo de gracia** y los días restantes. Después, Studio vuelve a ser la edición Learning.

Cuando una licencia termina

Vencer, cancelar o no pasar una comprobación solo devuelve Studio a la edición Learning:

- sus proyectos se abren igual que antes;
- el código G ya generado se puede seguir exportando y enviando a la máquina;
- las operaciones de pago esperan hasta que renueve.

Studio nunca borra sus archivos ni bloquea sus datos.

Actualizaciones de una compra perpetua

Una compra perpetua cubre las versiones publicadas en los 12 meses siguientes a la compra, o a una renovación de pago de las actualizaciones. Una versión posterior funciona como la edición Learning, y la licencia muestra **Actualizaciones vencidas**. Siga usando la última versión cubierta, o renueve un año de actualizaciones en la página de licencias. Vea Actualizar Studio.

Pasar una licencia a otro equipo

1. En la página de licencias, escriba el código de activación y elija "Mostrar mis equipos".
2. Elija "Liberar este equipo" junto al equipo que ya no usa.
3. Active la licencia en el equipo nuevo.

Puede liberar hasta tres equipos en 30 días. Para más, escriba a hello@twinwright.com.

Un hardware nuevo o un sistema reinstalado cambia la huella. Studio indica entonces **Esta licencia está vinculada a otro equipo.** Libere la entrada antigua y active de nuevo. Mientras tanto, Studio funciona como la edición Learning.

Código de activación perdido

Use **¿Perdió su código de activación?** en la página de licencias. Si pide una dirección de correo, escriba aquella con la que compró: cada licencia comprada con ella recibe un código nuevo por correo. Si le pide que nos escriba, escriba a hello@twinwright.com desde la dirección a la que se envió su código de activación; lo comprobamos y le enviamos un código nuevo. En ambos casos el código anterior deja de servir para activar, y los equipos ya activados siguen funcionando.

El reloj del equipo

Si el reloj se atrasa más de dos días, Studio indica **El reloj de este equipo marca una hora anterior a la última vez que Studio lo comprobó.** Cuenta la hora más tardía, así que ninguna licencia ni prueba recupera días. Ponga bien la fecha y la hora.

Studio guarda la última hora que vio en dos sitios. Cuenta la más tardía de las dos, así que borrar una de ellas no devuelve días.

Si una licencia o una prueba aparece como terminada antes de tiempo después de que el reloj se pusiera una vez en una fecha futura, ponga bien la fecha y la hora y escriba a hello@twinwright.com.

Para administradores: la línea de comandos

Con estas opciones, el programa no abre ninguna ventana. La carpeta es opcional; sin ella se usa la carpeta de licencias del equipo.

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

- `--licence-status` lista cada archivo de licencia y cada prueba, si es válido y por qué; `--json` da todos los campos.
- `--licence-request` muestra el código de solicitud de este equipo.
- `--licence-import` comprueba un archivo de licencia y lo copia en la carpeta de licencias.

En Linux, el programa se llama `twinwright-studio`. Los clientes de IA pueden leer el mismo estado con la herramienta MCP de solo lectura `licence_status`.

Qué envía Studio y qué conserva nuestro servidor: Privacidad y lo que envía Studio.

Sus datos y copias de seguridad

Studio guarda su trabajo en sus documentos y el estado de las máquinas en una carpeta propia. Este capítulo indica qué está dónde, qué incluir en la copia de seguridad y cómo pasar a otro equipo.

Qué está dónde

Dos carpetas contienen todo lo que es suyo:

- `Documentos\Twinwright Studio`: su trabajo;
- `%LOCALAPPDATA%\Twinwright\Studio`: el estado propio del programa.

Trabajos, programas y sus versiones

`Documentos\Twinwright Studio\programs\<program>\` contiene un archivo por cada versión de un programa (`v0001.nc`) y una nota de cuándo, por quién y por qué se guardó (`v0001.json`). El trabajo de CAM que generó un programa se versiona a su lado (`job-v0001.json`). Studio nunca los purga.

Herramientas > Ajustes... > Archivos permite poner la carpeta de programas en otro lugar (en el próximo inicio).

Informes

`Documentos\Twinwright Studio\Reports\`: informes de inspección (PDF y CSV), hojas de amarre (HTML) y paquetes de diagnóstico (ZIP).

Las máquinas y sus mundos

Cada máquina tiene un mundo en la carpeta de estado: la mesa, la pieza con sus cortes, las herramientas y la tabla de herramientas, los decalajes de origen, los ajustes del controlador, las temperaturas, las horas de funcionamiento y la calibración del palpador. Studio lo guarda mientras usted trabaja, cada vez que la máquina lleva dos segundos parada después de un cambio, y otra vez cuando usted cierra la ventana. El mundo de la primera máquina está en `world\`, los de las demás en `machines\<id>\world\`.

Ajustes

`settings.json`, en la carpeta de estado, contiene todo lo de **Herramientas > Ajustes...**; `window.json`, la posición y el tamaño de la ventana.

El registro de auditoría

`Documentos\Twinwright Studio\logs\` contiene un archivo por día (`audit-YYYY-MM-DD.jsonl`). Las entradas están encadenadas, de modo que un cambio o un hueco se nota al verificar el registro (**Herramientas > Ajustes... > Registro > Verificar integridad**). Por defecto el registro conserva 180 días y como máximo 2 GB; **Herramientas > Ajustes... > Registro** lo cambia. Copie la carpeta completa para que la cadena se conserve íntegra.

Qué incluir en la copia de seguridad

Cierre Studio y copie después las dos carpetas: `Documentos\Twinwright Studio` y `%LOCALAPPDATA%\Twinwright\Studio`.

Dos cosas no se trasladan con una copia:

- las claves de los servicios de IA: están cifradas para su cuenta de Windows en este equipo. Vuelva a escribirlas en el equipo nuevo;
- la licencia: pertenece al equipo. Actívela de nuevo en el equipo nuevo (consulte Licencias).

Pasar a otro equipo

1. En el equipo antiguo, cierre Studio y copie las dos carpetas.
2. Instale Studio en el equipo nuevo, inícielo una vez y ciérrelo.
3. Copie las dos carpetas en los mismos lugares del equipo nuevo.
4. Si en los ajustes movió la carpeta de programas o la del registro, compruebe que esas carpetas existan.
5. Inicie Studio, active la licencia y vuelva a escribir las claves de IA.
6. Vuelva a conectar sus clientes de IA (**IA > Conectar clientes de IA...**).

Empezar de nuevo con una máquina

Un restablecimiento de fábrica deja la máquina de su ventana tal como llegó: se elimina su mundo (la pieza, las herramientas, los decalajes, los ajustes del controlador, la calibración del palpador) y Studio se inicia de nuevo.

1. Elija **Archivo > Restablecer estado de fábrica...** (nivel Técnico y superior).
2. En un plazo de 5 segundos, vuelva a elegirlo para confirmar.

Solo se restablece esa máquina. Sus programas, informes, el registro de auditoría, los ajustes y las demás máquinas se conservan. El restablecimiento queda registrado en el registro de auditoría.

Actualizar Studio

Studio no busca versiones nuevas por sí mismo y nunca se actualiza solo. Usted decide cuándo actualizar.

Saber si hay una versión nueva

1. En Studio, elija **Ayuda > Acerca de**. Resultado: la línea bajo el nombre del programa muestra su versión, por ejemplo "Versión 1.0.0 · compilada el 2026-10-10 · commit 4f1c2ab".
2. Abra la página de descarga, twinwright.com/es/download. Resultado: la página indica la versión actual y la suma de comprobación SHA-256 del instalador.

Si las versiones no coinciden, la de la página de descarga es la más nueva.

Instalar una actualización

Necesita el instalador nuevo. No necesita permisos de administrador: Studio se instala para su cuenta de Windows.

1. Descargue el instalador desde la página de descarga.
2. Opcional: compare su suma de comprobación SHA-256 con la de la página. En PowerShell, ejecute `Get-FileHash` con el nombre del archivo.
3. Inicie el instalador. Si Windows dice que el editor es desconocido, no lo ejecute: escriba a hello@twinwright.com.
4. Si Studio está abierto, ciérrelo cuando el instalador se lo pida.
5. Siga el instalador. Resultado: la versión nueva sustituye al programa en la misma carpeta.

Sus ajustes, claves de IA, versiones de programa, registro de auditoría y licencia se quedan como están: no están en la carpeta del programa. Los clientes de IA que conectó siguen funcionando, porque la ruta del programa no cambia.

Para muchos equipos, el instalador funciona en modo silencioso: `setup.exe /S`. Vea Instalar Studio.

Qué versiones cubre su licencia

Licencia	Versiones con sus funciones de pago
Edición Learning	Todas las versiones
Suscripción	Todas las versiones, mientras dure la suscripción
Compra perpetua, propietario de MC-1	Las versiones publicadas en los 12 meses siguientes a la compra, o a una renovación de pago de las actualizaciones

Una versión publicada después de que terminen las actualizaciones de una licencia funciona como la edición Learning. La licencia muestra entonces **Actualizaciones vencidas**. Antes de actualizar, compruebe la fecha en **Ayuda > Licencia…**. Guarde el instalador de la última versión que cubre su licencia.

Para seguir actualizando, renueve un año de actualizaciones en la página de licencias. Vea Licencias, puestos y activación.

Equipos sin internet

1. En un equipo con internet, descargue el instalador.
2. Llévelo al equipo sin internet, por ejemplo en una memoria USB.
3. Instálelo allí como arriba.

Una versión más nueva no cambia el archivo de licencia. Renueve como antes: Activar un equipo sin internet.

Volver a una versión anterior

1. Haga una copia de seguridad de sus archivos antes de actualizar.
2. Desinstale Studio: en la Configuración de Windows, en "Aplicaciones".
3. Instale el instalador de la versión anterior.

> **Nota**
>
> Una versión anterior quizá no abra proyectos o ajustes guardados por una posterior. Conserve la copia de seguridad hasta estar seguro.

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Privacidad y lo que envía Studio

Studio trabaja en su equipo. Studio por sí mismo solo envía lo que necesitan el inicio de una prueba, la activación y la renovación de una licencia de pago. No hay telemetría, ni búsqueda de versiones nuevas, ni envío automático de los cierres por error. Si conecta el asistente de IA a un servicio en la nube, sus mensajes y lo que adjunte van a ese servicio: vea El asistente de IA y los servicios en la nube. Este capítulo indica qué sale, qué se queda en el equipo y cómo eliminarlo. El texto vinculante es la política de privacidad del sitio web, twinwright.com/es/privacy.

Qué envía Studio

Cuándo	A quién	Qué
Usted empieza la prueba de una edición de pago	El servidor de licencias de Twinwright, por HTTPS	La edición, un hash de la huella del equipo, la versión de Studio
Usted activa una licencia de pago	El servidor de licencias de Twinwright, por HTTPS	El código de activación, un hash de la huella del equipo, la versión de Studio
Studio renueva una licencia de pago (vea Renovación)	El servidor de licencias de Twinwright, por HTTPS	El número de licencia, el mismo hash, la versión de Studio

Eso es todo lo que Studio envía por sí mismo a Twinwright. La huella nunca sale del equipo tal cual; solo sale su hash. Un equipo activado sin conexión no envía nada.

Tres cosas solo salen si usted las configura:

- **El asistente de IA, conectado a un servicio en la nube.** Envía sus mensajes y lo que adjunte a ese servicio; vea más abajo.
- **Un cliente de IA que conecte por MCP**, como Claude o ChatGPT. El cliente envía lo que lee del gemelo a su propio servicio, según sus condiciones.
- **Una página web que abre Studio**, como la página de licencias. La abre su navegador; Studio no le añade nada.

El asistente de IA y los servicios en la nube

El asistente de IA trabaja con el proveedor que usted elija en él: un servicio en la nube como Anthropic, OpenAI o DeepSeek, con su propia cuenta y clave de API, o un modelo local.

Cuando envía un mensaje a un servicio en la nube, Studio envía a ese servicio, desde su equipo y con su clave:

- su mensaje, con la conversación anterior;
- la máquina, el amarre, el programa (con números de línea) y los problemas que encontraron las comprobaciones en la ventana, mientras **Adjuntar el programa, el amarre y los resultados de comprobación de la ventana (si han cambiado)** esté marcado. La casilla está marcada al principio. Studio los adjunta al primer mensaje y después solo cuando han cambiado;
- los resultados de las herramientas que el asistente usa en Studio, como la comprobación o la simulación de un programa, el estado de la máquina o entradas del registro, e imágenes de resultados para los servicios que aceptan imágenes.

Antes de que el primer mensaje vaya a un servicio en la nube, Studio muestra una vez lo que va a enviar.

El servicio trata todo esto según sus propias condiciones; Twinwright no recibe nada de ello. Para enviar menos, desmarque la casilla antes de enviar, o empiece de nuevo con **Nueva conversación**.

Con un modelo local (LM Studio, Ollama), todo esto se queda en el equipo.

En este equipo, el registro de auditoría guarda los primeros 180 bytes de cada mensaje (vea El registro de auditoría), y las claves de API se guardan cifradas (vea Qué conserva Studio en el equipo).

Qué conserva Studio en el equipo

Qué	Dónde (Windows)	Quién lo ve
Ajustes, el estado del gemelo, claves de IA	`%LOCALAPPDATA%\Twinwright\Studio`	Su cuenta de Windows
Versiones de programa	`Documentos\Twinwright Studio`	Su cuenta de Windows
El registro de auditoría	`Documentos\Twinwright Studio\logs`, salvo que los ajustes indiquen otra carpeta	Su cuenta de Windows
Informes de inspección, hojas de amarre, archivos de diagnóstico	`Documentos\Twinwright Studio\Reports`	Su cuenta de Windows
Informes de cierre inesperado	`%LOCALAPPDATA%\Twinwright\Studio\crash`, se borran a los 30 días	Su cuenta de Windows
Archivos de licencia, pruebas incluidas	`%ProgramData%\Twinwright\licence`	Todas las personas del equipo

Las claves de IA se guardan cifradas para su cuenta de Windows (Windows DPAPI). Las demás cuentas del mismo equipo no pueden leerlas.

El registro de auditoría

El registro de auditoría es un archivo por día, `audit-YYYY-MM-DD.jsonl`, en la carpeta de registros. Anota:

- lo que la IA y usted hicieron hacer a la máquina;
- los primeros 180 bytes de cada mensaje del panel de IA;
- las comprobaciones de licencia: el número de licencia, la edición y el resultado, sin el nombre del titular.

Las claves, los tokens y las contraseñas se escriben como `***`. En los ajustes se fija cuánto anota el registro y cuánto tiempo lo conserva. El registro se queda en el equipo; Studio nunca lo sube.

Si Studio se cierra de forma inesperada

En el momento del fallo, Studio escribe un informe breve: dónde falló, la secuencia de llamadas que llevó hasta allí y la lista de módulos del programa. No escribe contenido de la memoria.

En el siguiente inicio, Studio añade la información de la versión y las últimas 200 líneas de su registro, depuradas:

- las rutas y los nombres de archivo se sustituyen por marcadores;
- se quitan los nombres y las direcciones de correo;
- no hay código G, ni conversación con la IA, ni código de licencia, ni huella.

Después, Studio pregunta una vez. Puede abrir la carpeta del informe, guardar el informe como archivo o cerrar el mensaje. No se envía nada. Si quiere ayudarnos a encontrar la causa, envíe usted mismo el archivo a hello@twinwright.com.

Información de diagnóstico

Cuando nos pida ayuda, puede que le pidamos información de diagnóstico. En el menú **Ayuda**, elija "Recopilar información de diagnóstico". Studio guarda un archivo ZIP con los elementos marcados en la carpeta de informes:

Elemento	Marcado al principio
Versión, sistema, gráficos, idioma y unidades	Sí
Ajustes, sin las claves de IA	Sí
El perfil de la máquina y los ajustes `$` del controlador	Sí
Los últimos 7 días del registro y los informes de cierre inesperado, depurados como arriba	Sí
El estado de la licencia: edición y fecha de fin, sin códigos ni huella	Sí
El programa y el amarre actuales: los datos de su pieza	No

Abra y lea el archivo antes de enviarlo. No se envía nada automáticamente.

Qué escucha en la red

Los servicios de Studio para clientes de IA (MCP) y para programas de envío solo escuchan en este equipo (`127.0.0.1`). Otros equipos no pueden conectarse a ellos.

La opción de línea de comandos `--bind any` abre a su red los puertos de red del gemelo, como el de los programas de envío. El servicio MCP sigue siendo local.

> ⚠ **Advertencia**
>
> En modo sombra, el puerto de los programas de envío pasa las órdenes a la máquina real. Con `--bind any`, cualquiera de su red que llegue a ese puerto podría mover la máquina. Use `--bind any` solo para el gemelo y en una red que usted controle.

Qué conserva nuestro servidor

De cada licencia de pago, el servidor de licencias conserva:

- los hashes de un máximo de tres equipos por puesto, con la fecha en que se vio cada uno por primera y por última vez, para que pueda liberar uno en la página de licencias;
- el código de activación solo como hash: ni siquiera nosotros podemos leerlo;
- durante 90 días, un registro de pruebas, activaciones, renovaciones y cambios, con el hash del equipo, la versión de Studio y su dirección IP reducida a la parte de red.

De cada prueba conserva el hash del equipo con la edición y las fechas de la prueba, para que cada equipo tenga cada prueba una sola vez.

Nada de su trabajo llega al servidor.

Eliminar lo que conserva Studio

1. Libere el equipo en la página de licencias para que deje de contar (vea Pasar una licencia a otro equipo).
2. Desinstale Studio.
3. Borre las carpetas de la tabla de arriba que ya no necesite. Resultado: no queda nada de Studio en el equipo.

Al desinstalar se quita el programa, no sus archivos: sus proyectos, versiones de programa, registros e informes se quedan hasta que usted los borre.

Laboratorios, aulas y equipos compartidos

Este capítulo es para la persona que instala Studio en equipos que usan varias personas: en un laboratorio, un taller o un aula.

Una licencia para todo el equipo

La carpeta de licencias pertenece al equipo, no a una cuenta de Windows (vea La carpeta de licencias). Una licencia activada en un equipo vale para cualquiera que inicie sesión en él.

Un puesto de pago lo usa una sola persona a la vez, en un máximo de tres equipos, y un equipo que comparten varias personas cuenta una sola vez. Por eso:

- dé a un equipo compartido, como el que está junto a la máquina, un puesto propio;
- quienes también usen las funciones de pago en sus propios equipos necesitan puestos para ellos.

Active el equipo compartido una vez, con la cuenta de la persona que lo administra.

Lo que cada persona conserva

Studio se instala para cada cuenta de Windows. Cada persona tiene sus propios:

- ajustes y claves de IA: las claves están cifradas para esa cuenta y las demás cuentas no pueden leerlas;
- versiones de programa, informes y registro de auditoría, en su propia carpeta `Documentos\Twinwright Studio`.

La prueba de una edición cuenta una sola vez para el equipo, para todas las personas que lo usan: el servidor de licencias da a cada equipo una prueba de cada edición.

Para instalar en muchas cuentas y equipos, vea Instalar Studio.

Aulas

La edición Learning es gratuita en cada equipo, para aprender y enseñar. No hay que activar nada: instale Studio y funcionan todo el gemelo, el editor y el CAM para piezas corrientes.

Si puede, dé a cada estudiante su propia cuenta de Windows. Así sus ajustes, programas y registros quedan separados.

Para enseñar con las funciones de pago, los centros educativos y de formación pueden obtener puestos Pro. Escriba a hello@twinwright.com.

Equipos que se restauran en cada inicio

Algunos equipos de laboratorio vuelven a una imagen guardada en cada inicio, lo que también vacía la carpeta de licencias. En un equipo así:

1. Active el equipo una vez sin conexión (vea Activar un equipo sin internet) y conserve su archivo de licencia.
2. Ponga el archivo en la carpeta de licencias de la imagen, o impórtelo en cada inicio:

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

Un archivo de licencia está hecho para un solo equipo. Cada equipo del aula necesita su propio código de solicitud y su propio archivo.

Equipos sin internet

Actívelos sin conexión y renueve a mano: antes de que termine un periodo de suscripción, y una vez pasados los primeros 30 días de una compra perpetua. Vea Activar un equipo sin internet.

Cuando no se puede escribir en la carpeta de licencias

En Windows, cualquier cuenta puede crear `%ProgramData%\Twinwright\licence`. La cuenta que la crea puede modificar sus archivos; las demás pueden leerlos. Con eso basta: una persona importa la licencia y todos la usan.

Si las normas de su organización no lo permiten:

- un administrador crea la carpeta y da a los usuarios permiso de escritura; o
- la variable de entorno del sistema `TWINWRIGHT_LICENCE_DIR` indica una carpeta que todos pueden leer.

En Linux, la carpeta es `/etc/twinwright/licence` cuando su cuenta puede crearla o escribir en ella; si no, Studio usa su propia `~/.local/share/Twinwright/licence` (bajo `$XDG_DATA_HOME` si está definido), solo para usted. Para compartir una licencia entre los usuarios del ordenador, cree la carpeta una vez con `sudo install -d -m 1777 /etc/twinwright/licence`.

La IA en un equipo compartido

Cada persona conecta su propio proveedor de IA o cliente de IA, con su propia clave. El registro de auditoría de cada cuenta anota lo que esa persona y la IA hicieron hacer a la máquina. Vea Privacidad y lo que envía Studio.

Technical manual

Los modelos del gemelo y sus límites

El gemelo es el MC-1 como programa informático: el núcleo propio de Twinwright se ejecuta en él, y modelos físicos representan las partes de la máquina que rodean al controlador. Cada número que el gemelo da sobre el mundo físico es una estimación obtenida de un modelo físico. Sus parámetros proceden del diseño de la máquina, de las hojas de datos de las piezas y de valores publicados; cada MC-1 debe medirse en la puesta en marcha y su gemelo ajustarse a las mediciones (previsto). Cada sección siguiente termina con aquello en lo que se basan sus parámetros.

Borrador (ronda 14): el texto se está escribiendo; las secciones aún pueden crecer.

El controlador

El controlador del gemelo es el núcleo propio de Twinwright, compilado a partir del mismo código fuente que el firmware de la máquina. Lo que se modela es la placa que lo rodea: sus temporizadores y pulsos de paso, sus entradas con su antirrebote, su enlace con el equipo (el puerto serie USB y su protocolo de enlace) y el tiempo que consume el primer plano del controlador.

Se basa en: el código propio del núcleo; el tiempo del primer plano es una estimación hasta que se mida en la placa.

Movimiento y accionamientos

Cada eje lo mueven un accionamiento paso a paso de lazo cerrado y un husillo de bolas. El gemelo los modela con cuatro niveles de detalle; el predeterminado es un modelo de tres masas:

- el lazo de posición del accionamiento: el resorte magnético del motor en torno a la posición ordenada, su par que cae con la velocidad, la pérdida de sincronismo cuando la carga es demasiado alta, la alarma del accionamiento cuando el error de seguimiento crece demasiado;
- la fricción en las guías, la tuerca y los rodamientos, y la gravedad en Z;
- la flexibilidad de la transmisión entre el motor, el husillo de bolas y la mesa, el juego de la tuerca, el bastidor bajo el eje X;
- la alimentación de los accionamientos: la cadena de parada de emergencia, el contactor, el tiempo que tardan los accionamientos en ponerse en marcha.

Se basa en: el diseño (masas, inercias, rigidez por cálculo manual y por análisis de elementos finitos), las hojas de datos de los accionamientos y de los motores; la fricción y los parámetros internos del accionamiento son supuestos hasta que se midan.

Modelo de geometría y exactitud

Cuando está activado, el gemelo desplaza la punta de la herramienta según los errores geométricos de la máquina: el error de paso de cada eje y su parte periódica, dos errores de rectitud y tres errores angulares que actúan a través de sus brazos de Abbe, la perpendicularidad entre los ejes, el juego de la tuerca y la dilatación de los husillos de bolas y del husillo con el calor. Los finales de carrera ven los carros; el sensor de longitud de herramienta y el palpador ven la punta de la herramienta, como en la máquina. El modelo está desactivado por defecto: el gemelo nominal es ideal.

Se basa en: tablas de una máquina montada típica obtenidas de las clases de calidad de las piezas. No son la especificación de la máquina; se sustituyen por la medición de la máquina según ISO 230-2 en la puesta en marcha (previsto).

Calor

El husillo y los husillos de bolas se calientan con el uso y se enfrían en reposo; el gemelo conserva sus temperaturas entre sesiones y desplaza según ellas la punta de la herramienta y el paso de los husillos de bolas.

Se basa en: modelos térmicos de primer orden con parámetros supuestos hasta que se mida el calentamiento de la máquina.

Husillo y variador (variador de frecuencia)

El variador del husillo se modela como el de la máquina: el controlador se comunica con él por RS-485 en el protocolo propio del variador, byte a byte a la velocidad configurada, y el variador responde como describe su manual, incluidos los parámetros que deciden si escucha. Detrás de él: el par del motor a partir de su deslizamiento, su corriente y los disparos del variador por sobrecorriente, sobrecarga y subtensión, el bus de CC que mantiene en marcha el variador durante una breve caída de tensión, la bomba de agua.

Se basa en: los datos del variador y del motor; la capacitancia del bus, la parte magnetizante de la corriente y la fricción son supuestos hasta que se midan.

El corte

El bruto es un campo de alturas con regiones más finas donde usted las pide. Cada movimiento de la herramienta elimina lo que barre la forma de la herramienta, de modo que la forma de la pieza surge del propio programa.

La fuerza de corte procede de los coeficientes de corte del material, que cambian con el espesor de la viruta y la velocidad de corte, calculados diente a diente sobre la forma real de la herramienta (plana, esférica, toroidal, en V). A partir de la fuerza, el gemelo estima la carga y la potencia del husillo, la deflexión de la herramienta y el riesgo de rotura de la herramienta: la tensión de flexión en la raíz del filo frente a la resistencia del carburo.

Se basa en: datos de corte publicados para los materiales (la fuente de cada material está en el perfil de máquina), una herramienta afilada, una geometría de herramienta típica.

Vibración

Para cada corte, el gemelo calcula la profundidad a la que empezaría a vibrar con la velocidad del husillo en uso, a partir de los modos de vibración de la herramienta y de la máquina y de la geometría del corte, y elige una velocidad del husillo con la que el corte es estable. También estima dónde quedan las paredes a partir de la vibración forzada del corte.

La punta de la herramienta tiene tres modos en cada dirección: la estructura de la máquina, el portaherramientas en la pinza y el husillo, y la flexión propia de la herramienta. La frecuencia del modo de flexión depende de la forma de la herramienta (sus filos, su cuello, su mango y su voladizo): unos 14 kHz para una fresa de 6 mm con 20 mm de voladizo, más de 100 kHz para una de 0.2 mm. Las herramientas de menos de 1 mm aproximadamente vibran, si lo hacen, con su propia flexión, a una profundidad que apenas depende de la velocidad del husillo; el gemelo calcula esa profundidad en el dominio del tiempo para cada corte.

Se basa en: los modos del bastidor obtenidos por análisis de elementos finitos, un modo supuesto del portaherramientas y la flexión de la herramienta estimada a partir de su geometría con la teoría de vigas; la amortiguación de los tres es supuesta (3 %). Ninguno se ha medido todavía: deben medirse con una prueba de impacto en la máquina construida y, hasta entonces, las profundidades estables son estimaciones. La amortiguación del propio corte no se modela: para herramientas pequeñas, el índice de vibración queda del lado seguro.

Descentramiento

Una fresa cuyo eje está descentrado respecto al husillo corta con su diente más exterior: los agujeros salen más grandes y los resaltes más pequeños (hallazgo 187). El gemelo corta con esa envolvente; CAM lo tiene en cuenta en los ajustes. Una herramienta puede llevar su propio descentramiento en su especificación (la clase de una pinza de precisión o un valor medido; `tir3`: su TIR en micrómetros); si no, se usa el de la máquina.

Se basa en: un descentramiento supuesto hasta que se mida en la máquina.

Desgaste de la herramienta

En la 1.0 toda herramienta es nueva: la forma que corta y sus fuerzas son las de una herramienta afilada, por mucho que haya cortado. Contar la vida útil que un programa consume de cada herramienta está previsto para una versión posterior. Las herramientas que cortan aluminio suelen cambiarse antes, sobre todo porque el aluminio se pega al filo y la pieza muestra rebabas o un acabado peor, y no porque el flanco se haya desgastado.

El palpador

El palpador es una herramienta cuya bola hace conmutar la entrada del palpador cuando toca la pieza. El gemelo calcula el contacto de forma exacta a partir de la forma del bruto y de las superficies cortadas, incluidas las pendientes. Dos modelos: un palpador ideal que conmuta en cuanto la bola toca, y un palpador típico con recorrido previo que depende de la dirección, tres lóbulos, dispersión y rotura del vástago cuando se empuja demasiado.

Se basa en: valores típicos de un palpador de disparo por contacto; calibre el palpador en un anillo patrón antes de medir con él.

El espacio libre en Z

Para el amarre que está en la mesa, el gemelo calcula el punto más alto que hay que salvar y, para la herramienta del husillo y cada herramienta de la tabla de herramientas, la altura de la punta de la herramienta en el extremo superior de Z. Una herramienta que no puede salvar el amarre se notifica cuando cambia el amarre o la herramienta (hallazgo 188).

Se basa en: el diseño de la máquina y las longitudes de las herramientas tal como se indican.

Colisiones y sujeción de la pieza

El gemelo comprueba la herramienta, su portaherramientas y el husillo frente a la pieza, el utillaje y la máquina, y avisa antes de una colisión inminente. La sujeción (bridas, mordaza, prisma en V, cinta, una pieza fija) retiene la pieza con una fuerza; un corte que empuja con más fuerza puede hacer que se deslice.

Se basa en: la geometría de la máquina y de los utillajes tomada del diseño; las fuerzas de sujeción son estimaciones.

Lo que no se modela

- la textura y la rugosidad de la superficie más allá de la posición de las paredes;
- el filo recrecido, las virutas y su evacuación;
- las paredes delgadas que se flexionan bajo el corte;
- el calor de la propia pieza;
- lo que ocurre después de que se rompa una herramienta.

Las pruebas de aceptación que necesitan una máquina construida (las mediciones de los accionamientos y del sonido, las fuerzas de corte, la superficie, el calor, los modos del bastidor, el modo sombra con una máquina real) están en las notas de diseño del gemelo (docs/06, sección 7).

El núcleo propio de Twinwright

El núcleo propio de Twinwright es el firmware de controlador CNC de Twinwright. En Studio hace funcionar la máquina del gemelo. El mismo código se compila para la placa controladora del MC-1.

Usa el protocolo serie de Grbl 1.1, de modo que los programas de envío de código G funcionan con él. Los ajustes, las alarmas y los errores conservan los números que los programas de envío conocen.

Qué hace

- **Movimiento:** planificación con anticipación sobre 100 bloques, velocidades en las esquinas a partir de `$11` y arcos sobre el círculo real. El movimiento se ejecuta en intervalos de 1 ms. Cada eje respeta su velocidad máxima (`$110`–`$112`) y su aceleración (`$120`–`$122`).
- **Funciones de la máquina:**
  - la referencia, los límites por hardware y por software, el palpado y el sensor de longitud de herramienta;
  - el variador del husillo (variador de frecuencia) por Modbus y la refrigeración;
  - los cambios de herramienta según los define `$341`;
  - la puerta de seguridad con estacionamiento;
  - los ajustes (override), el desplazamiento manual y el volante electrónico.
- **Programas en el controlador:** los programas de su sistema de archivos se ejecutan con subprogramas de palabras O y macros (consulte Palabras O).
- **Código G:** RS274/NGC y LinuxCNC. Incluye una tabla de herramientas, la compensación del radio de herramienta, la mezcla de trayectorias, expresiones y parámetros (consulte Código G y código M).

El protocolo serie, los comandos en tiempo real y los comandos `$` están en Protocolo. Cada ajuste está en Ajustes, cada alarma en Códigos de alarma y cada error en Códigos de error.

En comparación con otros controladores Grbl

Los programas de envío escritos para Grbl funcionan con el controlador. Esta sección enumera lo que se nota cuando se está acostumbrado a otro controlador Grbl.

En qué casos se detiene el controlador

En estos casos algunos controladores Grbl siguen adelante; el núcleo propio de Twinwright se detiene o genera una alarma.

- **El variador del husillo se detiene por sí solo** (un disparo, una sobrecarga o su propia tecla de parada): el avance se detiene en 0.2 s como máximo y el controlador genera `ALARM:14`.
- **El variador del husillo no responde:** los ejes frenan primero hasta detenerse, de modo que se conserva la posición. Después el controlador genera `ALARM:19`.
- **Cambio de herramienta:**
  - El husillo siempre se detiene para un cambio de herramienta.
  - Si un cambio de herramienta continúa sin confirmación, el husillo y la refrigeración vuelven a como estaban.
- **Reanudación tras una retención de avance o la puerta:** el husillo alcanza su velocidad antes de que la herramienta se mueva.
- **La puerta vuelve a abrirse mientras la máquina regresa del estacionamiento:** la máquina se detiene y vuelve a subir a la altura de estacionamiento. Espera al siguiente inicio de ciclo.
- **Se pierde el programa de envío** (se desconecta el USB, se cierra el puerto): el trabajo se detiene. `$1030` lo controla; está activado por defecto.
- **Interruptores del palpador y de referencia:** los pulsos espurios en los cables no cuentan como un contacto.
- **El texto UTF-8 en un programa** (chino, "Ø", "–") se lee como texto, nunca como un comando en tiempo real.
- **Un arco en formato R sin cuerda en su plano**, por ejemplo `G91 G2 X0 Z1 R5`, responde con `error:33`.

Más código G

Algunos controladores Grbl rechazan o ignoran lo siguiente; el núcleo propio de Twinwright lo ejecuta como lo definen los estándares:

- la tabla de herramientas (`G43 Hn`, `G43.2`, `G10 L1/L10/L11`);
- la compensación del radio de herramienta (`G41`, `G42`);
- la mezcla de trayectorias (`G64 P`) y la parada exacta (`G61.1`);
- los subprogramas de palabras O y el control de flujo, los parámetros `#` y las expresiones;
- `G52`, las coordenadas polares (`@`, `^`) y `M99`;
- `G85`, `G86` y `G89`, de taladrado y mandrinado, donde algunos controladores Grbl los aceptan y no se mueven;
- lo que suelen escribir los posprocesadores de CAM: una línea `%` con un nombre, `G49 H0`, líneas de número de programa Fanuc (`O1000`) y el `M25` de Centroid.

Más fiel al programa

- **Los arcos** siguen el círculo real. Algunos controladores Grbl los cortan en cuerdas que se apartan del círculo hasta 4 × `$12`.
- **Los arcos en tiempo inverso** (`G93`) duran el tiempo programado. Algunos controladores Grbl los ejecutan varias veces más rápido.
- **Los tiempos de espera** (`G4`) duran lo programado. Algunos controladores Grbl los redondean a pasos de 50 ms.
- **Taladrado con descarga:** cada agujero recibe los pasos de descarga que necesita. Cuando la profundidad es un número entero de pasos, algunos controladores Grbl taladran un paso completo más desde la altura de retirada.
- **Los arcos grandes cerca de los límites por software** no generan falsas alarmas.

- `M72` restablece la compensación de longitud de herramienta que guardó `M70` o `M73`. Algunos controladores Grbl conservan una compensación fijada después cuando la guardada era 0 con el mismo modo de compensación (`G43.1`), así que después de `M72` el mismo programa puede funcionar allí con otra compensación de longitud de herramienta.
- **Aceleración:** cada eje respeta `$120`–`$122`, también en los arcos pequeños. Un programa con una gran cantidad de arcos pequeños puede tardar hasta alrededor de un 2 % más que en algunos controladores Grbl. Un programa normal tarda alrededor de un 0.1 % menos.

Detalles del protocolo

- **Informes de estado:**
  - En una retención de avance o con la puerta abierta, el avance en `FS:` es 0: los ejes están parados.
  - La distancia restante se informa hasta que los ejes se detienen.
  - Durante una alarma crítica, `Bf:` muestra el planificador vacío.
  - Con el estado del analizador activado en `$10`, `[GC:]` llega una vez por cambio.
- **Comprobaciones de entrada:**
  - Un número de ajuste incorrecto, como `$45x`, responde con `error:3`.
  - Un ajuste entero escrito como `$1=25.0` guarda 25; algunos controladores Grbl guardan 250.
  - `M6` y un movimiento en una misma línea responden con `error:40`.
  - `G65` o `G66` con otros comandos en una misma línea responden con `error:21`.
- **Alarmas y arranque:**
  - No hay una pausa de 100 ms después de una alarma.
  - El primer arranque con el almacenamiento de ajustes vacío no responde con `error:7`.
- **Textos y ajustes:**
  - Las descripciones de los ajustes, las alarmas y los errores son propias de Twinwright.
  - El controlador tiene ajustes propios: `$1030` (detenerse cuando se pierde el programa de envío) y `$1000` en adelante para el input shaping (desactivado por defecto).
- `$39=0` (caracteres en tiempo real heredados desactivados): `?`, `!` y `~` son texto dentro de las líneas `$` y de los comentarios, y comandos en cualquier otro lugar. El controlador lo decide a partir de los bytes a medida que llegan. Por eso responde a `?` también con la puerta abierta o en una retención de avance, donde algunos controladores Grbl a veces no lo hacen.

No admitido

- `G65 P7` (mensajes Modbus).
- Las variables de sistema de Fanuc. `#4120` y similares son parámetros normales y se leen como 0.
- El roscado (`G33`, `G76`, `G33.1`) sin un encoder de husillo. El MC-1 no tiene ninguno, de modo que responden con `error:20`.
- Los ciclos de desbaste de torno (otros controladores Grbl tampoco los tienen).
- Red, drivers Trinamic y otras marcas de variador del husillo, que algunos controladores Grbl añaden como plug-ins.

Programas de envío

Los programas de envío que se han comprobado, y sus ajustes, están en Programas de envío de código G. Para comparar una máquina real con su gemelo, consulte Máquina real y modo sombra.

Código G y código M

El núcleo propio de Twinwright ejecuta el código G que se indica a continuación. Los códigos siguen RS274/NGC y LinuxCNC. Un código que el controlador no admite responde `error:20`, y la línea no hace nada.

Para los comandos en tiempo real (bytes individuales como `?`, `!` y `~`), consulte Protocolo. Para saber en qué se diferencia el controlador de otros controladores Grbl, consulte El núcleo propio de Twinwright.

Código G

Código	Significado	Notas
`G0`, `G1`	Movimiento rápido, avance lineal	
`G2`, `G3`	Arco en sentido horario, en sentido antihorario (I J K o R), hélice	La herramienta sigue el círculo real. Un arco en forma R sin cuerda en su plano responde `error:33`.
`G4 P`	Tiempo de espera, P segundos	El tiempo de espera dura lo programado.
`G5`, `G5.1`	Spline cúbico, spline cuadrático	Un `G5` sin I ni J continúa en la dirección en que terminó el movimiento anterior.
`G7`, `G8`	Modo de diámetro de torno, modo de radio	Solo en modo torno (`$32=2`).
`G10 L2`, `G10 L20`	Definir un decalaje de origen	
`G10 L1`, `G10 L10`, `G10 L11`	Definir la tabla de herramientas	32 herramientas. La tabla se conserva cuando el controlador está apagado.
`G17`, `G18`, `G19`	Seleccionar el plano	
`G20`, `G21`	Unidades en pulgadas, unidades en milímetros	
`G28`, `G30`	Ir a una posición guardada pasando por un punto intermedio	
`G28.1`, `G30.1`	Guardar la posición actual	
`G33`, `G33.1`, `G76`	Movimiento sincronizado con el husillo, roscado rígido, ciclo de roscado	Requieren un codificador en el husillo. El MC-1 no tiene ninguno, de modo que responden `error:20`.
`G38.2`–`G38.5`	Palpado	El resultado es el punto en que se disparó el palpador, notificado como `[PRB:]`. Fije el cero a partir de ese punto, no de donde se detuvieron los ejes.
`G40`	Compensación del radio de herramienta desactivada	
`G41`, `G42`, `G41.1`, `G42.1`	Compensación del radio de herramienta: D de la tabla de herramientas, o `.1` con un diámetro	Solo en el plano G17. Movimientos de entrada y salida como en LinuxCNC. Se rechaza un movimiento que produciría un sobrecorte.
`G43 Hn`, `G43.2 Hn`	Compensación de longitud de herramienta de la tabla de herramientas, compensación añadida	
`G43.1`	Compensación dinámica de longitud de herramienta	
`G49`	Compensación de longitud de herramienta desactivada	
`G50`, `G51`	Escalado desactivado, activado	
`G52`	Sistema de coordenadas local	Comparte su decalaje con `G92`, como en LinuxCNC.
`G53`	Coordenadas de máquina para esta línea	
`G54`–`G59`, `G59.1`–`G59.3`	Sistemas de coordenadas de trabajo	`$486` puede bloquear `G59.1`–`G59.3`.
`G61`	Trayectoria exacta	
`G61.1`	Parada exacta	Los ejes se detienen al final de cada movimiento programado.
`G64 P Q`	Suavizado de trayectoria	Donde ahorra tiempo, la esquina entre dos movimientos rectos se convierte en un arco a menos de P de la esquina. Q se acepta y no se usa.
`G65 P1`–`P6`	Macros integradas	Ajustes, compensaciones de herramienta, parámetros, estado de la máquina, elección del palpador, espera del husillo. `P7` (mensajes Modbus) no se admite.
`G65 P100` y superiores, `G66`, `G66.1`, `G67`	Macros de archivo, llamadas modales a macros	P debe ser un entero. Ningún otro comando puede compartir la línea (`error:21`).
`G73`, `G81`, `G82`, `G83`	Ciclos de taladrado, taladrado con descarga	Cada agujero recibe las descargas que necesita.
`G84`, `G85`, `G86`, `G89`	Ciclos de roscado con macho y de mandrinado	
`G80`	Ciclo desactivado	
`G90`, `G91`	Distancias absolutas, distancias incrementales	
`G91.1`	Centros de arco incrementales	
`G92`, `G92.1`, `G92.2`, `G92.3`	Decalaje de coordenadas: definir, borrar, suspender, restaurar	
`G93`, `G94`, `G95`	Tiempo inverso, unidades por minuto, unidades por revolución	Los arcos en tiempo inverso tardan el tiempo programado.

Código	Significado	Notas
`G96`, `G97`	Velocidad de corte constante, velocidad de husillo constante	Solo en modo torno.
`G98`, `G99`	Retirada del ciclo a la altura inicial, al plano R	

Estos códigos G no se admiten (`error:20`):

- `G5.2`, `G5.3` (NURBS);
- `G17.1`–`G19.1` (planos UVW);
- `G70`–`G72` (ciclos de torno);
- `G90.1`.

Código M

Código	Significado	Notas
`M0`, `M1`	Pausa del programa, parada opcional	El comando en tiempo real `0x88` activa o desactiva la parada opcional (`M1`).
`M2`, `M30`	Fin del programa	
`M3`, `M4`, `M5`	Husillo en sentido horario, en sentido antihorario, apagado	
`M6`	Cambio de herramienta	`$341` determina lo que hace M6. M6 y un movimiento en una misma línea responden `error:40`.
`M7`, `M8`, `M9`	Refrigeración: por niebla, por chorro, apagada	En el MC-1, M8 activa la refrigeración por agua del husillo.
`M25`	Z hacia arriba hasta la altura de referencia	Código de Centroid, que escriben algunos posprocesadores de CAM.
`M48`, `M49`	Ajustes (override) activados, desactivados	
`M50`–`M53`	Ajuste de avance, ajuste del husillo, avance adaptativo, retención de avance: activados o desactivados	
`M56`	Estacionamiento activado o desactivado	
`M60`	Pausa para el cambio de palé	
`M61`	Fijar la herramienta actual sin hacer un cambio	
`M62`–`M65`	Salida digital activada o desactivada, con el siguiente movimiento o de inmediato	En placas con salidas auxiliares.
`M66`	Esperar una entrada	La espera comienza cuando han terminado los movimientos anteriores.
`M70`–`M73`	Guardar y restaurar el estado modal	
`M98 P L`, `M99`	Llamar a un subprograma, volver	En programas que se ejecutan desde el sistema de archivos del controlador. Un `M98` enviado línea a línea responde `error:39`. Un `M99` en el programa principal lo termina.

Estos códigos M no se admiten (`error:20`):

- `M19` (orientación del husillo);
- `M67`, `M68` (salidas analógicas; el MC-1 no tiene ninguna);
- `M100`–`M199` (códigos M de usuario).

Palabras, sintaxis y parámetros

Estructura de la línea

- Se admiten los comentarios `( … )` y `;`.
- `(MSG,…)` envía el texto al programa de envío como mensaje.
- `(PRINT,…)` envía un mensaje con los valores de los parámetros que nombra.
- `(DEBUG,…)` hace lo mismo mientras `#5599` está activado.
- Se admiten el salto de bloque `/`, los números de línea `N` y las líneas `%` (con o sin nombre).
- Las mayúsculas y minúsculas no importan, y los espacios dentro de las palabras se ignoran.

Parámetros y expresiones

- Los parámetros numerados `#1`–`#5000` pertenecen al programa. Se leen como 0 hasta que el programa los define.
- `#5001` y superiores, y los parámetros de sistema con nombre, solo se pueden leer.
- Parámetros con nombre `#<name>`: leer un nombre que nunca se definió responde `error:39`. `EXISTS[#<name>]` indica si se definió.
- Las expresiones entre `[ ]` usan los operadores y las funciones de LinuxCNC. Los ángulos están en grados.
- Un valor que define una línea no se puede leer en esa misma línea.

Palabras O

- Los programas que se ejecutan desde el sistema de archivos del controlador pueden usar subprogramas, `if`, `elseif`, `else`, `while`, `do` y `repeat`.
- Los subprogramas y los bucles enviados línea a línea se recopilan y se ejecutan cuando están completos.
- Un subprograma también puede ser un archivo propio: `<name>.ngc` o `<name>.macro`.

Coordenadas polares

`@` es el radio y `^` el ángulo, en el plano XY alrededor del origen de pieza.

Errores en una línea

- Dos códigos del mismo grupo modal en una línea responden `error:21`.
- Dos comandos que usan ambos las palabras de eje en una línea responden `error:24`.
- La lista completa de códigos de error está en Códigos de error.

Estrategias y parámetros de CAM

Qué hace cada estrategia, qué significan sus parámetros y cómo fija el gemelo los datos de corte. Los números que da el gemelo son estimaciones de sus modelos físicos.

Los parámetros siguientes son los que un elemento admite en un trabajo (los elementos de la herramienta `program_generate`). Las longitudes están en milímetros y los ángulos en grados.

Vaciar una cajera: anillos, zigzag y adaptativo

El área de una cajera es su forma menos sus islas, desplazada hacia dentro el radio de la herramienta. Se vacía capa a capa, ninguna capa más profunda que la profundidad de corte (`ap`). La estrategia (`strategy`) fija la trayectoria dentro de una capa:

Estrategia	Trayectoria	Úsela para
`rings` (por defecto)	Desfases del área, desde su punto más interior hasta la pared, a `ae` uno de otro	La mayoría de las cajeras
`zigzag`	Pasadas rectas de ida y vuelta con el ángulo `angle`, a `ae` una de otra	Áreas abiertas y poco profundas
`adaptive`	Una trayectoria que mantiene el contacto de la herramienta cerca de `ae` en todos los puntos	Cajeras profundas, materiales más duros

Una cajera con ajuste acaba sus paredes a medida en una pasada propia. Consulte Ajustes.

El desbaste adaptativo corta con un contacto radial pequeño y una `ap` mayor. Sus enlaces se desplazan sin tocar el material, y recurre a los anillos donde el área es demasiado estrecha para él.

Una pasada de acabado envuelve la herramienta mucho más en una esquina interior de la pared que a lo largo de una pared recta. Donde la envuelve más de 1,2 veces, esa parte de la pared se corta primero por escalones (`corner_steps`, activado por defecto; desactivado cuando la operación da su propio `finish_ap`, salvo que también dé `corner_steps: true`; `corner_steps: false` lo desactiva).

Mecanizado de restos

Con `rest: true`, una cajera vacía solo lo que dejaron sin cortar las herramientas anteriores: las esquinas y las partes estrechas a las que una herramienta mayor no pudo llegar. Studio calcula lo que queda a partir de las cajeras anteriores del mismo programa, o a partir de la pieza de trabajo del gemelo cuando este la tiene.

Contornos y puentes

Un contorno (`side`: `outside`, `inside` o `on`) sigue un perfil capa a capa, en fresado en concordancia. `outside` y `inside` desplazan el perfil el radio de la herramienta; `on` lo sigue tal cual.

Los puentes (`tabs`: número, ancho y alto) dejan en el fondo uniones cortas de material, de modo que una pieza recortada se mantenga en su sitio. Un puente mide 1 mm de alto salvo que indique su altura; donde el corte tiene 1 mm de profundidad o menos, su altura es la mitad de la profundidad.

Ranuras: capas y trocoidal

Una ranura (`x0`, `y0`, `x1`, `y1`, `width`) se corta con la estrategia:

- `layers` (por defecto): el centro de la ranura con el ancho completo de la herramienta, por capas, y después hacia fuera hasta el ancho de la ranura a cada lado;
- `trochoidal`: pequeños bucles a lo largo de la ranura, cada uno `step` más adelante. La herramienta solo está en contacto a lo largo de un arco corto, por lo que puede cortar más profundo en una sola pasada. La ranura debe ser más ancha que la herramienta.

Cómo entra una herramienta en el material: hélice, rampa, penetración

Una fresa de mango no penetra en vertical en material macizo. Cada capa comienza con:

- una hélice, donde el área tiene espacio para una (su diámetro y su ángulo figuran en el `entry` de la respuesta);
- una rampa a lo largo de la trayectoria, donde no lo tiene;
- una penetración recta solo en un agujero o fuera del bruto.

La respuesta indica, para cada operación, cómo entró al principio.

Agujeros

Ciclos de taladrado

Los agujeros en `points`, de profundidad `depth`, se taladran con los ciclos del controlador. Un agujero más profundo que tres veces el diámetro de la broca se taladra con descarga, a razón de 1.5 veces el diámetro por descarga, salvo que `peck` indique otra. El plano de retirada del ciclo está 1 mm por encima de la superficie superior.

Mandrinado por interpolación helicoidal

Un agujero de diámetro `d` mayor que la fresa de mango se corta siguiendo una hélice y después se acaba a medida en el fondo. El agujero puede tener como máximo el doble del diámetro de la herramienta; los agujeros mayores se vacían como cajera (`shape: circle`).

Escariado

Un escariador (`reamer 6 H7`) del ajuste del agujero entra con su avance y sale con el mismo avance, sin salir del agujero en rápido. El agujero debe estar taladrado antes en el mismo trabajo, con 0.05–0.6 mm menos de diámetro, o estar marcado como `predrilled`. La velocidad del escariador es la cuarta parte de la velocidad de corte de la fresa de mango para el material, y su avance por revolución, 2.5 veces el de la broca.

Fresado de roscas, también de roscas cónicas para tubería

Una fresa de roscar baja por el agujero y después corta la rosca de abajo arriba siguiendo una hélice de un paso por vuelta: rosca a derechas en sentido antihorario (fresado en concordancia), rosca a izquierdas en sentido horario. La rosca se indica por su nombre (`thread`: `M8`, `M8x1`, `1/4-20 UNC`, `G1/4`) o mediante `d` y `pitch`. Las roscas cónicas para tubería (`1/4 NPT`, `Rc1/4`) se cortan siguiendo su cono, con cada vuelta en 48 arcos.

Ranuras anulares en un agujero

`groove` corta una ranura anular en la pared de un agujero (para un anillo elástico o una junta tórica, o una trampa de luz) con una fresa para ranuras en T: un disco sobre un cuello más fino. La fresa baja por el centro del agujero hasta la ranura, se acerca hasta 0,1 mm de la pared y da una vuelta completa (en concordancia) por cada `step` dentro de la pared; después vuelve al centro y sube. El trabajo da el diámetro de la ranura `d`, el del agujero `bore`, el borde inferior de la ranura `depth` y su altura `width` (por defecto, el grosor del disco; una ranura más alta se corta en varias pasadas de abajo arriba). Se rechaza cuando la herramienta del husillo no es una fresa en T, cuando el disco no baja por el agujero con 0,2 mm de holgura, cuando la ranura es más profunda de lo que el disco sobresale de su cuello y cuando la ranura no es mayor que el agujero.

Chaflanes, grabado y tallado en V

- Chaflán: una fresa en V sigue el borde del perfil a la profundidad que da el ancho `width`. El ancho está limitado por el cono.
- Grabado: la herramienta sigue líneas, o texto en una fuente de un solo trazo o de contorno, a una única profundidad `depth`.
- Tallado en V: la fresa en V sigue el eje medial de la forma. Su profundidad en cada punto da el ancho allí, de modo que las esquinas de la forma salen vivas. `max_depth` aplana las zonas anchas a esa profundidad.

Para grabar sobre una superficie curva, indique `surface`: el nombre de una malla de `geometry_import`. Cada trazo se muestrea cada 0,2 mm, la herramienta se apoya en la malla y baja `depth` por debajo de donde la toca; en una pendiente, una fresa en V toca primero en el lado más alto, así que en ningún punto se corta más hondo que `depth`. Un trazo fuera de la malla se rechaza.

El perlage, el motivo de círculos solapados de los relojeros, es `perlage`: una fresa plana tan ancha como un círculo baja hasta `depth` (0,2 mm como máximo) en cada punto de una retícula dentro de un contorno, espera `dwell` segundos (0,2 por defecto) y sube 0,3 mm por encima de la cara para ir al siguiente. Las filas se desplazan medio `pitch` (por defecto 0,65 veces el diámetro de la herramienta), y cada círculo cubre parte del anterior. `margin` mantiene los centros de los círculos a esa distancia dentro del contorno.

Desbaste y acabado 3D

Una malla se desbasta por niveles (`rough3d`), con cada nivel vaciado de forma adaptativa. Después se acaba (`finish3d`) con:

Estrategia	Trayectoria
`parallel`	Pasadas en una dirección sobre la pieza
`waterline`	Contornos de nivel, para paredes empinadas
`steep_shallow` (por defecto)	Z constante en las zonas empinadas, paralela en las poco inclinadas
`pencil`	A lo largo de las esquinas interiores, después de las demás

`rest_from` acaba solo lo que dejó una herramienta anterior.

Desbaste de restos: con `rest_from` (la fresa más grande, plana o tórica, que desbastó antes, por su designación o su número de herramienta), `rough3d` corta solo lo que esa herramienta dejó, como los huecos entre nervios a los que no llegaba; `rest_allowance` es la creces que dejó (por defecto, la de esta pasada).

Datos de corte del gemelo

Studio parte de los datos del material para el tipo de herramienta: el intervalo de velocidades de corte y el avance por diente. Los valores que usted indique en `cutting` (`rpm`, `feed`, `ap`, `ae`, `finish_ap`) se mantienen tal como los da. Los demás valores se fijan con los modelos del gemelo.

Estabilidad y vibración

El modelo de vibración del gemelo calcula los lóbulos de estabilidad de la herramienta en su portaherramientas: la profundidad de corte que se mantiene estable a cada velocidad del husillo. Los lóbulos solo están donde los sitúa el modelo de la herramienta y de la máquina, así que Studio no apunta al pico de un lóbulo: para cada velocidad toma la profundidad estable más pequeña dentro de un 10 % alrededor de ella, elige una velocidad entre 0.7 y 1.8 veces la que da el material y corta como mucho el 80 % de esa profundidad.

Potencia, fuerza y deflexión de la herramienta

- Potencia: el corte consume como máximo alrededor del 80 % de la potencia del husillo a esa velocidad.
- Fuerza: la fuerza de corte se mantiene por debajo de la mitad de lo que el utillaje sujeta lateralmente.
- Deflexión: una pasada de acabado se mantiene lo bastante ligera para que la estimación del gemelo sobre la deflexión de la herramienta siga siendo pequeña.
- Esquinas y arcos: el avance en un arco está limitado por la aceleración de que dispone la máquina, para que el controlador no reduzca la velocidad por debajo del avance con el que se planificó el corte. Las pasadas de acabado reducen la velocidad en las esquinas interiores de una cajera, donde crece el contacto de la herramienta.

El bucle cerrado: probar el programa en el gemelo

Con la verificación activada, Studio ejecuta el programa en un gemelo nuevo. Cuando la ejecución muestra vibración, una carga demasiado alta o una deflexión excesiva de la herramienta, Studio modifica el programa y lo prueba de nuevo:

- capas menos profundas,
- otra velocidad del husillo, entre 0.7 y 1.8 veces la ajustada, en la que el corte sigue siendo estable aunque los lóbulos estén algo desplazados,
- pasadas de repaso en las paredes de un ajuste.

Cada versión probada se guarda. Un programa largo se prueba en segundo plano, después de la llamada, y su informe llega más tarde (`run_status`). La respuesta indica qué cambió y por qué.

Varios amarres y anidado

- Amarres: Studio escribe el programa de cada amarre y después comprueba los amarres en orden en el gemelo, cada uno sobre la pieza de trabajo que dejaron los amarres anteriores. Donde esa pieza de trabajo difiere del bruto, el programa del amarre se escribe de nuevo para ella.
- Anidado: las piezas se colocan en la placa según sus perfiles, con un margen entre ellas. Cada pieza conserva sus elementos, que giran y se mueven con ella.

Límites

- Tres ejes. Sin cuarto eje y sin inclinación de la herramienta.
- Los archivos STEP no se leen; exporte STL o 3MF desde el programa de CAD.
- Los datos de corte y las comprobaciones del gemelo son estimaciones obtenidas de modelos físicos de la máquina, la herramienta y el material. Una herramienta o un material que difiere de su modelo corta de otra manera.
- Ediciones: el desbaste adaptativo, las estrategias de acabado 3D distintas de `parallel` , el mecanizado de restos y los varios amarres están en la edición Maker y superiores; el anidado y el optimizador de programas están en la edición Pro.

Qué tamaño puede tener un trabajo

Más allá de estos límites, Studio no escribe el programa. Indica por qué y qué debe cambiarse.

Qué	Límite
Amarres en un trabajo	8
Anidado	16 tipos de pieza, 200 copias de cada una
Una malla (STL, 3MF, OBJ, glTF)	5 millones de triángulos; el archivo, hasta 1 GB
Un archivo SVG, DXF o de texto	64 MB; un dibujo DXF, hasta 2 millones de segmentos
Desbaste adaptativo	La cuadrícula del recorrido admite unos 1.5 millones de celdas. Una cajera del tamaño del recorrido del MC-1 se vacía con una fresa de mango de 3 mm; con una de 1 mm, use anillos o zigzag, o un paso lateral mayor.

Un trabajo puede tener hasta 32000 elementos; los agujeros iguales van como un solo elemento con sus puntos. La lista de operaciones, los motivos y las reparaciones del bucle cerrado abarcan los primeros 64 elementos; un elemento posterior que vibra se notifica, pero no se repara. La hoja de amarre enumera las primeras 64 operaciones y lo indica.

La comprobación del programa y la hoja de amarre leen hasta 2 millones de movimientos. Un arco cuenta como sus cuerdas cortas: unas 50 para un círculo completo de 3 mm de radio, unas 220 para uno de 50 mm. Pasado ese límite, indican dónde se detuvieron. La hoja de amarre sigue dando el tiempo de todo el programa, pero no los tiempos de las operaciones ni las alturas de las herramientas a partir de esa línea. La prueba en el gemelo comprueba el programa completo.

Cuánto tarda Studio en escribir un programa, medido en dos núcleos de un PC de sobremesa (la prueba en el gemelo viene después):

Trabajo	Tiempo	Memoria
20 cajeras con islas en una placa de 180 x 130 mm	menos de 1 s	18 MB
Una cajera del tamaño del recorrido, adaptativa, fresa de mango de 6 mm	12 s	47 MB
Lo mismo con una fresa de mango de 3 mm	37 s	70 MB
1000 caracteres grabados	12 s	25 MB
200 arandelas anidadas en una placa	5 s	30 MB
Desbaste 3D de un relieve del tamaño del recorrido (medio millón de triángulos), fresa de mango de 6 mm	8 min	240 MB

Trabajo	Tiempo	Memoria
El mismo relieve con 3 millones de triángulos	20 min	1 GB
Acabado 3D paralelo del mismo relieve, fresa esférica de 2 mm, altura de cresta de 0.01 mm (medio millón de triángulos)	4 min	270 MB
Lo mismo con 3 millones de triángulos	18 min	1 GB

Para el trabajo 3D:

- Haga la malla solo tan fina como la necesite la pieza. El acabado tarda más aproximadamente en la misma proporción en que la malla tiene más triángulos.
- En una superficie suavemente ondulada, acabe con `parallel` o `steep_shallow`, no con `waterline`. Z constante hace allí muchos bucles pequeños: en el relieve anterior se movió más en el aire que en el material, y su programa tuvo 2.8 millones de líneas.

Ajustes, descentramiento y métodos de inspección

Cómo corta Studio una medida a un ajuste, cómo tiene en cuenta el descentramiento de la fresa y cómo mide una pieza con el palpador. Los números del gemelo que figuran aquí son estimaciones de sus modelos físicos; Studio no declara ninguna precisión de posicionamiento para la máquina.

Ajustes (ISO 286)

Una medida puede llevar un ajuste: el diámetro de un agujero (`"d": "Ø12 H7"`), el de un eje (`"fit": "g6"` en un contorno exterior), el ancho de una ranura o las medidas X e Y de un rectángulo. Studio toma las desviaciones límite del ajuste de las tablas de la ISO 286-2 para esa medida, o las desviaciones límite tal como usted las escriba (`"tol": "+0.015/0"`, en milímetros).

Cortar al punto medio de la zona

La pared se desbasta dejando sobremetal y después se acaba, en una pasada propia, hasta el punto medio de la zona del ajuste. Los ajustes de grado IT6 y más finos reciben una pasada de repaso: la vuelta de acabado de nuevo, a medida, para quitar lo que dejó la deflexión de la herramienta. El bucle cerrado añade pasadas de repaso donde la estimación del gemelo sobre la deflexión las requiere.

La respuesta indica, para cada ajuste, la zona, el diámetro al que se cortó y el motivo.

Ajustes habituales

Nombre	Par	Uso
`sliding`	H7/g6	Gira o desliza, juego pequeño
`locating`	H7/h6	El ajuste con juego más pequeño: se monta a mano
`transition`	H7/k6	Un poco de juego o un poco de apriete: para posicionar, a presión ligera
`press`	H7/p6	Con apriete: se introduce a presión

Un par nombra las dos piezas: el agujero toma el primero y el eje el segundo.

El descentramiento de la fresa

Una fresa que gira con descentramiento corta un agujero mayor que ella misma. El gemelo modela el descentramiento de la herramienta en su portaherramientas como una envolvente. Para una medida con ajuste, Studio desplaza la trayectoria de acabado, hacia dentro en un agujero y hacia fuera en un eje, en el valor de esa envolvente, de modo que la medida que estima el gemelo siga siendo el punto medio de la zona. Un agujero escariado no se modifica: es la medida del escariador.

La envolvente es la estimación del gemelo para su modelo de la herramienta y del portaherramientas. Mida la primera pieza.

Medir con el palpador

Un toque y su compensación

Cada toque es una secuencia de movimientos de palpado (G38):

1. un movimiento protegido hasta cerca de la superficie, que se detiene si el palpador se dispara por el camino;
2. un palpado rápido hasta la superficie;
3. un movimiento de retroceso hasta que el palpador se libera, y un poco más;
4. un palpado lento hasta la superficie: solo se usa el punto de disparo de este;
5. un movimiento de vuelta al punto de partida.

El punto de disparo es el centro de la bola. Studio suma el radio efectivo de la bola en la dirección del toque. El radio efectivo y la lobulación del palpador proceden de su calibración en un anillo patrón de diámetro conocido.

Agujeros, salientes, paredes y fondos

- Agujeros y salientes: cuatro u ocho toques alrededor de la pared a una profundidad. Studio ajusta un círculo a los centros de la bola por mínimos cuadrados y después suma el radio efectivo.
- Paredes: toques perpendiculares a la pared, en puntos a lo largo de ella.
- Fondos y caras superiores: toques en vertical hacia abajo.
- Las posiciones se dan en las coordenadas del programa, desde el origen de pieza.

Tolerancias generales (ISO 2768-1)

Una medida sin tolerancia propia adopta la tolerancia general de la clase solicitada (f, m, c, v), según la tabla de la ISO 2768-1 para su intervalo de medidas: medidas lineales, y las alturas y los ángulos de los chaflanes.

Cuando la cuadrícula del gemelo es demasiado gruesa para una tolerancia, se indica que la estimación del gemelo para esa comprobación es demasiado gruesa para juzgarla; el valor medido se juzga de todos modos.

Comprobaciones de forma (ISO 2768-2, primer paso)

La clase H, K o L tras la clase de la ISO 2768-1 (por ejemplo “mK”) añade comprobaciones de forma, juzgadas según las tablas de la ISO 2768-2:

Comprobación	Toques	Tabla
Planitud	Un fondo en una cuadrícula de 3 × 3; la dispersión de los puntos respecto de su plano	1
Rectitud	Una pared larga, de 3 a 8 puntos a lo largo de ella; la dispersión respecto de su recta	1
Perpendicularidad	Una pared a una segunda profundidad; su inclinación a lo largo de la altura de la pared, respecto del eje Z de la máquina	2
Circularidad	Un agujero o saliente en 8 puntos; la dispersión de los radios	4 (la tolerancia del diámetro)

Dos toques en una pared solo pueden distinguirse cuando están lo bastante lejos uno de otro. Una pared demasiado corta para ello, por ejemplo bajo un chaflán de su borde, se enumera con el motivo en lugar de un valor.

Un palpador calibrado solo en su radio efectivo tiene una lobulación propia. Se refleja en la circularidad: el informe indica de qué tamaño la considera el modelo de palpador del gemelo.

Chaflanes y ranuras en V

- Un chaflán se toca dos veces a lo largo de su pendiente, a 0.3 y 0.7 de su ancho desde el perfil. La recta que pasa por los dos centros de la bola es paralela a la pendiente: da el ancho, la profundidad y el ángulo del chaflán.
- Una ranura en V (un grabado o un tallado en V) se toca dos veces en cada flanco. Las dos rectas dan la profundidad de la ranura en su vértice, su ancho en la parte superior, su ángulo incluido y su desplazamiento respecto de la línea a lo largo de la cual se cortó.
- Una rosca no se mide. Sus crestas son el agujero recto, y el palpador del gemelo solo encuentra la parte superior de las celdas; la respuesta lo indica.

Qué significa la estimación del gemelo

El gemelo corta su pieza de trabajo simulada con el mismo programa y después realiza en ella los mismos toques con el mismo modelo de palpador. Su estimación es lo que da ese cálculo. Muestra el efecto del propio programa sobre cada medida, hasta donde lo modela el gemelo: la deflexión de la herramienta, su descentramiento, las fuerzas de corte.

La estimación no es la incertidumbre de medición de la máquina. El informe no dice nada sobre con qué precisión mide o posiciona la máquina.

Límites

- El palpador mide lo que toca. Un elemento más estrecho que la bola, o una superficie a la que la bola no puede llegar perpendicularmente, no se mide; la respuesta indica por qué.
- La pieza de trabajo del gemelo es una cuadrícula. Sus estimaciones solo son tan finas como esa cuadrícula; donde es demasiado gruesa para una tolerancia, la estimación del gemelo no se juzga.
- Las comprobaciones de forma son un primer paso de la ISO 2768-2: planitud, rectitud, perpendicularidad y circularidad. La simetría y el paralelismo aún no se comprueban.

Perfiles de máquina

Borrador (ronda 14): el texto se está escribiendo; las secciones aún pueden crecer.

Qué es un perfil de máquina

Un perfil de máquina es el archivo a partir del cual el gemelo construye su máquina: un archivo por modelo de máquina, en JSON. Contiene la geometría de la máquina y sus ejes, los accionamientos y los husillos de bolas, el husillo y su variador, los sensores y su cableado, la sujeción de la pieza, los datos de corte de cada material, el sensor de longitud de herramienta y el palpador, los sonidos y con qué se construye el controlador. Los ajustes del controlador vienen con él como un archivo de ajustes.

El perfil registra de dónde proceden sus valores (su procedencia): el diseño, una hoja de datos, una fuente publicada, un supuesto o una medición.

De dónde procede

El perfil del MC-1 se genera a partir del diseño paramétrico de la máquina: los mismos parámetros que producen sus planos y su lista de materiales producen su perfil, de modo que el gemelo y el diseño de la máquina no pueden divergir. El perfil conserva las sumas de comprobación de los archivos de diseño a partir de los que se hizo. Un esquema comprueba su forma; el capítulo de referencia Esquema del perfil de máquina enumera todas las claves.

Los valores de su máquina

Cada MC-1 debe medirse en la puesta en marcha y su gemelo ajustarse a las mediciones (previsto; el procedimiento y sus herramientas están escritos y probados en el gemelo):

- los finales de carrera: dónde conmutan, su histéresis y su repetibilidad;
- cada eje según ISO 230-2: el juego, la tabla de errores de paso, los pasos por milímetro;
- los coeficientes de corte a partir de ranuras de prueba;
- la inercia y la fricción del husillo a partir de una parada por inercia;
- el palpador en un anillo patrón.

Los valores medidos se registran con la fecha, la herramienta que los midió y cómo; el perfil se reconstruye con ellos, cada uno marcado como medido y con el valor del diseño conservado en su historial, y los ajustes del controlador que cambian se envían a la máquina.

Varias máquinas

Studio puede mantener varias máquinas de un modelo. Cada una conserva su propio mundo entre sesiones (dónde se encuentra, su pieza, sus herramientas, la memoria de su controlador) y su propia disposición de ventana; el perfil del modelo de máquina es el mismo para todas hasta que se mide una máquina.

La referencia

Cada clave del perfil, con su tipo, su unidad y el valor del MC-1: Esquema del perfil de máquina.

Vista previa: las indicaciones de seguridad de esta página aún no las ha revisado un hablante nativo.

Hardware del MC-1

Borrador (ronda 14): el texto se está redactando; las secciones aún pueden crecer.

El MC-1 tal como está diseñado: las cifras de este capítulo son valores de diseño. Cada máquina construida debe medirse en la puesta en servicio, y su gemelo debe ajustarse a lo medido (consulte Perfiles de máquina).

La máquina

El MC-1 es una fresadora de sobremesa de tres ejes para aluminio, latón y plásticos técnicos. Su pórtico es fijo; la mesa se mueve en Y, el carro del pórtico en X y el husillo en Z.

Recorridos X / Y / Z	200 / 150 / 80 mm
Mesa	300 × 220 × 35 mm, agujeros roscados M6 en una cuadrícula de 25 mm
Tamaño, ancho × fondo × alto	unos 508 × 490 × 564 mm, con motores, cadenas portacables y patas de nivelación, sin la caja de control
Masa	unos 57 kg, sin la caja de control
Bastidor	placas de aluminio 6061-T6, atornilladas
Guías lineales	carriles HIWIN HGR15 (6), bloques de brida HGW15CC (12)
Husillos de bolas	SFU1204 (12 mm, paso de 4 mm), C7, sobre soportes BK10 / BF10
Motores	motores paso a paso de lazo cerrado NEMA 23, 1.2–1.3 N·m (3), con acoplamientos de garras
Husillo	800 W, refrigerado por agua, Ø65 × 195 mm, pinza ER11
Caja de control	una caja aparte junto a la máquina, a su derecha

Ejes y accionamientos

Cada eje es un motor paso a paso de lazo cerrado que mueve su husillo de bolas directamente mediante un acoplamiento de garras. Los accionamientos notifican un fallo por una sola línea: las salidas de alarma de los tres accionamientos están cableadas en serie con la entrada de fallo de motor del controlador, de modo que un accionamiento que falla, o que pierde su alimentación, detiene la máquina con una alarma.

Los movimientos más rápidos del diseño son 3000 mm/min en X e Y y 1500 mm/min en Z.

Finales de carrera y referencia

Cinco interruptores inductivos M8 marcan los extremos del recorrido: uno en cada extremo de X y de Y, cada uno en su propia entrada, y uno en la corredera de Z para ambos extremos de Z. Los interruptores son normalmente cerrados: un cable roto o un conector suelto se lee como un interruptor accionado, de modo que el controlador activa una alarma y se niega a hacer la referencia en lugar de ir a chocar contra el tope.

La referencia se hace primero en Z, después en X e Y, cada eje hacia su extremo positivo (X+, Y+, Z+). Hay que hacer la referencia de la máquina después del encendido.

Z y la gravedad

La corredera de Z lleva el husillo, de unos 6.35 kg. Mientras los accionamientos reciben alimentación, mantienen Z. Cuando no la reciben (tras una parada de emergencia, un fallo de un accionamiento o un corte de alimentación), Z con el husillo puede descender lentamente.

> ⚠ **Advertencia**
>
> Tras una parada de emergencia o un fallo de un accionamiento, vuelva a hacer la referencia de la máquina antes de continuar. No meta la mano bajo el husillo mientras los accionamientos estén apagados.

La puesta en servicio incluye una prueba con los accionamientos apagados: si Z desciende lentamente, la máquina recibe un motor de Z con freno o un contrapeso.

Husillo

El husillo es un electrohusillo de 800 W refrigerado por agua, de Ø65 × 195 mm con pinza ER11, dos polos, hasta 24 000 rpm. Un variador Huanyang HY01D523B (VFD, entrada monofásica de 220 V) lo hace funcionar; el controlador fija su velocidad y lee su estado por RS-485, a través de un módulo de aislamiento. Una bomba de agua refrigera el husillo.

- El relé de fallo del variador también está cableado a la línea de fallo de motor: cuando el variador se dispara, el controlador detiene el programa con una alarma en lugar de hacer avanzar una herramienta detenida contra la pieza.
- En una parada de emergencia, el variador pierde su alimentación junto con los accionamientos, y el husillo se detiene por inercia (no tiene freno).

Controlador y cableado

El controlador es un Teensy 4.1 sobre una placa portadora T41U5XBB, con entradas optoaisladas. Ejecuta el núcleo propio de Twinwright, compatible con el protocolo serie de Grbl 1.1, y se conecta al equipo por USB.

Entrada	Qué hay en ella
Finales de carrera	X−, X+, Y−, Y+ y Z, cinco entradas
Palpado	el sensor de longitud de herramienta y el palpador
Parada de emergencia	el segundo contacto del pulsador de parada de emergencia
Fallo de motor	las salidas de alarma de los tres accionamientos y el relé de fallo del variador del husillo, en serie
Inicio de ciclo, retención de avance	los pulsadores SB2 y SB3 de la caja de control

Alimentación

La red eléctrica pasa por un interruptor automático. Una fuente de 24 V está siempre encendida: alimenta el controlador, los sensores y el circuito de control, de modo que el controlador sigue conectado al equipo y notifica la parada de emergencia. Un contactor (KM1) conmuta la alimentación de 48 V de los accionamientos, el variador del husillo y la bomba.

Los pulsadores de la caja de control

Cuatro pulsadores de Ø22 en la puerta de la caja de control, vistos de frente:

	izquierda	derecha
fila superior	SB1 encendido (verde)	SB0 parada de emergencia (rojo)
fila inferior	SB2 inicio de ciclo (verde)	SB3 retención de avance (amarillo)

- **SB1 encendido** cierra KM1, que después se mantiene por sí solo. Tras una parada de emergencia o un corte de alimentación, la máquina no vuelve a recibir alimentación por sí sola: libere la parada de emergencia y vuelva a pulsar SB1.
- **SB3 retención de avance** hace lo mismo que la retención de avance (`!`) de su programa de envío: los ejes se detienen de forma controlada y el husillo sigue girando. Está debajo de la parada de emergencia, de modo que una mano que no acierta con la parada de emergencia pulsa la retención de avance, no el inicio de ciclo.
- **SB2 inicio de ciclo** continúa, igual que el inicio de ciclo (`~`) de su programa de envío.

> **Precaución**
>
> La retención de avance no es una función de seguridad. Para detener la máquina por seguridad, use la parada de emergencia.

Sensor de longitud de herramienta y palpador

El sensor de longitud de herramienta está en la parte trasera izquierda de la mesa, sujeto por dos tornillos M6 en la cuadrícula; ocupa la esquina del área de trabajo en aproximadamente X 0–57, Y 130–150 mm. El sensor de longitud de herramienta y un palpador comparten la entrada de palpado del controlador.

Con Z en la parte superior, la punta de una herramienta que sobresale 15.5 mm de la pinza está a 80 mm sobre la mesa, y la nariz del husillo a 130 mm. La punta de una herramienta más larga queda más baja en su longitud adicional. Un palpador sujeto en una pinza ER suele medir de 60–90 mm desde la tuerca hasta la punta del stylus: con Z en la parte superior, un palpador así solo salva amarres bajos, unos 20 mm sobre la mesa con un palpador de 70 mm.

Studio lo comprueba por usted: cuando prepara la pieza, cambia una herramienta o edita la tabla de herramientas, compara el punto más alto del amarre con la punta de cada herramienta con Z en la parte superior e indica qué herramienta no pasa con holgura y en cuánto.

Sujeción de la pieza

Los agujeros M6 de la mesa, en una cuadrícula de 25 mm, admiten bridas, una subplaca o una mordaza. El gemelo conoce una mordaza autocentrante de 75 mm sobre una subplaca de aluminio de 10 mm, calzos paralelos, bridas planas, cinta de doble cara y un bloque en V.

- Con la mordaza sobre su subplaca, la parte superior de las garras queda a unos 66 mm sobre la mesa, de modo que una pieza sujeta en la mordaza alcanza unos 71 mm de altura o más.
- Para las piezas que quiera palpar, use un amarre bajo: bridas planas sobre la subplaca (la cara superior del bruto a 15–30 mm sobre la mesa) o una mordaza baja. Un palpador no cabe por encima de una pieza sujeta en la mordaza de 75 mm.
- Quitar la subplaca de 10 mm da 10 mm más de espacio a las herramientas; para un palpador no basta.

Seguridad

- **Parada de emergencia (SB0)**: abre el circuito de la bobina del contactor, que desconecta los accionamientos, el variador del husillo y la bomba, y avisa al controlador mediante su segundo contacto. Los ejes y el husillo pierden la alimentación; el husillo se detiene por inercia; Z puede descender lentamente (consulte Z y la gravedad). Liberar la parada de emergencia no reinicia nada: pulse SB1, espere a que arranquen los accionamientos, reinicie el controlador y haga la referencia.
- **Fallos**: un fallo de un accionamiento, un disparo del variador del husillo o un cable roto de un final de carrera detiene la máquina con una alarma.

- **Retención de avance** es una parada controlada para el trabajo, no para la seguridad.
- El MC-1 no tiene cerramiento. Manténgase alejado del husillo mientras gira y mientras los ejes pueden moverse, use protección ocular y monte fuelles o cubiertas de cepillo sobre los husillos de bolas de X y Z contra las virutas.
- Ninguna de las funciones de seguridad de la máquina usa IA, y el asistente de IA de Studio no es una función de seguridad.

Qué se mide en la puesta en servicio

Los valores de diseño anteriores deben comprobarse en cada máquina construida, y su gemelo debe ajustarse a lo medido:

- el punto de disparo de cada final de carrera (los interruptores inductivos barridos lateralmente pueden dispararse unos 2 mm más tarde de lo dibujado), y después el retroceso de la referencia y los límites por software;
- las salidas de alarma de los accionamientos: desenchufar una, o desconectar los 48 V, debe producir una alarma;
- la prueba de Z con los accionamientos apagados;
- los parámetros del variador del husillo, un arranque desde el panel del variador sin carga y después el control por RS-485;
- la caja de control: la parada de emergencia y SB1 como se ha descrito arriba, cada final de carrera y el sensor de longitud de herramienta desenchufados uno por uno (cada uno debe producir una alarma) y los pulsadores del panel de mando.

Cómo toma el gemelo los valores medidos: Perfiles de máquina.

Reference

Esta referencia se genera a partir del programa y solo está en inglés.

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`.

Referencia de ajustes

Todos los ajustes del controlador. `$<n>` muestra uno, `$<n>=<value>` lo cambia y `$$` los lista todos. Los valores indicados son los de la MC-1. Un ajuste marcado con *reinicio* surte efecto después de reiniciar el controlador.

Índice

- General
- Señales de control
- Límites
- Refrigeración
- Husillo
- Cambio de herramienta
- Búsqueda de referencia
- Palpado
- Estacionamiento/Puerta de seguridad
- Desplazamiento manual
- Modbus
- Motores paso a paso
- Variador de frecuencia
- Eje X
- Eje Y
- Eje Z

General

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
`$10`	Opciones del informe de estado		campo de bits		15871	Lo que contiene el informe de estado en tiempo real. El estado del analizador se envía en su propia línea, solo cuando ha cambiado. Bits: 0 Posición en coordenadas de máquina, 1 Estado del búfer, 2 Números de línea, 3 Avance y velocidad, 4 Estado de los pines, 5 Decalaje de coordenadas de pieza (WCO), 6 Ajustes (override), 7 Coordenadas de palpado, 8 Sincronizar el búfer al cambiar el WCO, 9 Estado del analizador, 10 Subestado de alarma, 11 Subestado de ejecución, 12 Habilitar durante la búsqueda de referencia, 13 Distancia por recorrer
`$11`	Desviación en esquinas	mm	decimal		0.002	Desviación máxima que puede tener la trayectoria en una esquina entre dos movimientos; fija la velocidad en la esquina. Cuanto menor, más lento y más exacto.
`$12`	Tolerancia de arcos	mm	decimal		0.002	La exactitud con que se siguen los arcos.
`$13`	Informar en pulgadas		booleano		0	Las posiciones y los avances de los informes están en pulgadas.
`$28`	Distancia de retracción de G73	mm	decimal		0.100	La retracción corta entre las descargas de G73.
`$32`	Modo de funcionamiento		opción		0	Fresadora, láser o torno. Opciones: 0 Normal, 1 Modo láser, 2 Modo torno
`$39`	Habilitar comandos en tiempo real heredados		booleano		1	«?», «!» y «~» actúan como comandos en tiempo real en todas partes, también dentro de los comandos $ y de los comentarios.
`$60`	Restablecer los ajustes (override)		booleano		0	Los ajustes (override) vuelven al 100 % al final de un programa.
`$62`	Habilitar suspensión		booleano		0	Pasar a suspensión tras un tiempo largo en reposo.
`$63`	Acciones de la retención de avance		campo de bits		3	Lo que hace una retención de avance además de detener el movimiento. Bits: 0 Desactivar el láser durante la retención, 1 Restablecer el estado del husillo y de la refrigeración al reanudar, 2 Poner las rpm al mínimo (excepto M4 del láser)
`$64`	Forzar la alarma de inicio		booleano		0	Arrancar en estado de alarma tras el encendido.
`$384`	Desactivar la persistencia de G92		booleano		0	Los decalajes G92 no se conservan tras apagar y volver a encender.
`$398`	Bloques del búfer del planificador *(reinicio)*		entero	30–1000	100	El número de movimientos sobre los que mira por adelantado el planificador.
`$481`	Intervalo de informes automáticos *(reinicio)*	ms	entero	100–1000	0	Enviar informes de estado con este intervalo sin que se soliciten (0: desactivado).
`$484`	Desbloqueo necesario tras la parada de emergencia		booleano		1	Tras una parada de emergencia, la máquina permanece bloqueada hasta que se desbloquea ($X) o se referencia.
`$486`	Bloquear sistemas de coordenadas		campo de bits		0	Sistemas de coordenadas que G10 no puede cambiar. Bits: 0 G59.1, 1 G59.2, 2 G59.3

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$676	Acciones del reinicio		campo de bits		15	Lo que borra un reinicio. Bits: 0 Borrar el estado de referenciado si se perdió la posición, 1 Borrar los decalajes (excepto G92), 2 Borrar el ajuste de rápidos, 3 Borrar el ajuste de avance
$1030	Parar el trabajo al perder el host		booleano		1	Cuando la conexión del programa de envío que ejecuta un trabajo queda interrumpida durante 0.1 s (USB desconectado, puerto cerrado), el trabajo se detiene como con una parada (0x19): el movimiento decelera hasta detenerse, el husillo y la refrigeración se apagan, las líneas que envió y que aún no se han ejecutado se descartan y la posición se conserva. Cuando la conexión se restablece, se avisa al programa de envío. Un programa que se ejecuta desde el sistema de archivos sigue adelante sin su programa de envío.

Señales de control

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$14	Invertir entradas de control		campo de bits		0	Entradas de control que están invertidas. Bits: 1 Retención de avance, 2 Inicio de ciclo, 6 Parada de emergencia, 8 Fallo del motor
$17	Desactivar pull-up en entradas de control		campo de bits		0	Entradas de control sin la resistencia pull-up interna. Bits: 1 Retención de avance, 2 Inicio de ciclo, 6 Parada de emergencia, 8 Fallo del motor

Límites

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$5	Invertir entradas de final de carrera		máscara de ejes		7	Entradas de final de carrera que están invertidas, p. ej. para finales de carrera normalmente cerrados.
$18	Desactivar pull-up en entradas de final de carrera		máscara de ejes		0	Entradas de final de carrera sin la resistencia pull-up interna.
$20	Habilitar límites por software		booleano		1	Rechazar el movimiento más allá del recorrido ($130 ...). Requiere la búsqueda de referencia, para que se conozca la posición.
$21	Habilitar límites por hardware		campo de bits		1	Parar de inmediato cuando se acciona un final de carrera. Modo estricto: ningún movimiento mientras haya un final de carrera accionado. Bits: 0 Habilitar, 1 Modo estricto (el bit 0 habilita el resto)

Refrigeración

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$15	Invertir salidas de refrigeración		campo de bits		0	Salidas de refrigeración que son activas a nivel bajo. Bits: 0 Refrigeración por chorro, 1 Refrigeración por niebla
$673	Retardo de arranque de la refrigeración	s	decimal	0.5–20	0.0	El tiempo que se da a la refrigeración para arrancar tras M7/M8.

Husillo

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$30	Velocidad máxima del husillo	rpm	decimal		24000.000	La velocidad más alta del husillo.
$31	Velocidad mínima del husillo	rpm	decimal		6000.000	La velocidad más baja del husillo.
$340	Tolerancia de velocidad alcanzada del husillo	%	decimal		5.0	El movimiento tras M3/M4 espera hasta que la velocidad del husillo esté dentro de este porcentaje de la programada (0: sin espera).
$394	Retardo de arranque del husillo	s	decimal	0.5–60	15.0	El tiempo que se da al husillo para alcanzar la velocidad tras M3/M4 (con $340: la espera máxima antes de una alarma).
$539	Retardo de parada del husillo	s	decimal	0.5–60	0.0	El tiempo que se da al husillo para detenerse tras M5.

Cambio de herramienta

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$341	Modo de cambio de herramienta		opción		3	Lo que hace M6. Opciones: 0 Normal, 1 Medición manual de herramienta, 2 Medición manual de herramienta en G59.3, 3 Medición automática de herramienta en G59.3, 4 Ignorar M6
$342	Distancia de palpado en el cambio de herramienta	mm	decimal		70.0	La distancia hasta la que se busca el sensor de longitud de herramienta.

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$343	Avance de localización en el cambio de herramienta	mm/min	decimal		25.0	El avance lento que localiza con exactitud el sensor de longitud de herramienta.
$344	Avance de búsqueda en el cambio de herramienta	mm/min	decimal		200.0	El avance rápido con el que se busca el sensor de longitud de herramienta.
$345	Avance de retroceso del palpador en el cambio de herramienta	mm/min	decimal		200.0	El avance con el que se aleja del sensor de longitud de herramienta.
$346	Opciones de cambio de herramienta		campo de bits		0	Opciones del cambio de herramienta. Bits: 0 Restablecer la posición tras M6, 1 Cambiar la herramienta en G30, 2 Retroceso rápido del palpador
$485	Conservar el número de herramienta tras un reinicio		booleano		0	El número de la herramienta que hay en el husillo se conserva tras apagar y volver a encender.

Búsqueda de referencia

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$22	Ciclo de búsqueda de referencia		campo de bits		541	El ciclo de búsqueda de referencia y sus opciones. Bits: 0 Habilitar, 1 Habilitar comandos de un solo eje, 2 Búsqueda de referencia obligatoria al arrancar, 3 Fijar el origen de la máquina en 0, 4 Dos finales de carrera comparten una entrada, 5 Permitir manual, 6 Anular los bloqueos, 9 Avances por eje, 10 Ejecutar las líneas de inicio solo al completar la búsqueda de referencia (el bit 0 habilita el resto)
$23	Invertir dirección de la búsqueda de referencia		máscara de ejes		0	Ejes que buscan la referencia hacia su extremo negativo.
$26	Retardo antirrebote del final de carrera en la búsqueda de referencia	milisegundos	entero		250	El tiempo de espera tras accionarse un final de carrera, antes del siguiente movimiento de búsqueda de referencia.
$27	Distancia de retroceso del final de carrera en la búsqueda de referencia	mm	decimal		2.000	La distancia que retrocede la búsqueda de referencia al separarse del final de carrera.
$43	Pasadas de búsqueda de referencia		entero	1–128	1	El número de veces que se localiza el punto de accionamiento del final de carrera.
$44	Ejes de la primera fase de búsqueda de referencia		máscara de ejes		4	Los ejes que se referencian primero.
$45	Ejes de la segunda fase de búsqueda de referencia		máscara de ejes		3	Los ejes que se referencian en segundo lugar.
$46	Ejes de la tercera fase de búsqueda de referencia		máscara de ejes		0	Los ejes que se referencian en tercer lugar.

Palpado

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$6	Invertir entradas del palpador		campo de bits		1	Invierte la entrada del palpador, p. ej. para un sensor de longitud de herramienta normalmente cerrado. Bits: 0 Palpador
$19	Desactivar pull-up en entradas del palpador		campo de bits		0	La entrada del palpador sin la resistencia pull-up interna. Bits: 0 Palpador
$65	Opciones de palpado		campo de bits		0	Cómo se trata el movimiento de palpado. Bits: 0 Permitir el ajuste de avance, 1 Aplicar los límites por software

Estacionamiento/Puerta de seguridad

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$41	Ciclo de estacionamiento		campo de bits		0	Retraer el husillo cuando se abre la puerta de seguridad. Bits: 0 Habilitar, 1 Desactivar al iniciar, 2 Habilitar el control de anulación del estacionamiento (el bit 0 habilita el resto)
$42	Eje de estacionamiento		opción		2	El eje que realiza el estacionamiento. Opciones: 0 X, 1 Y, 2 Z
$56	Distancia de retirada lenta del estacionamiento	mm	decimal		5.0	La distancia de la primera retirada lenta de la herramienta.

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$57	Avance de retirada lenta del estacionamiento	mm/min	decimal		100.0	El avance de esa retirada.
$58	Destino del estacionamiento	mm	decimal	≥ -100000	-5.0	La posición a la que va el eje de estacionamiento, en coordenadas de máquina.
$59	Avance rápido del estacionamiento	mm/min	decimal		500.0	El avance del resto del movimiento de estacionamiento.
$61	Opciones de la puerta de seguridad		campo de bits		0	Lo que permanece activado mientras la puerta de seguridad está abierta. Bits: 1 Mantener el estado de la refrigeración al abrir la puerta
$392	Retardo de arranque del husillo	s	decimal	0.5–60	4.0	El tiempo que se da al husillo para alcanzar la velocidad cuando la puerta se vuelve a cerrar.
$393	Retardo de arranque de la refrigeración	s	decimal	0.5–20	1.0	El tiempo que se da a la refrigeración para arrancar cuando la puerta se vuelve a cerrar.

Desplazamiento manual

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$40	Limitar los comandos de desplazamiento manual		booleano		0	Los desplazamientos manuales se detienen en los límites por software en lugar de rechazarse.

Modbus

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$374	Velocidad en baudios de Modbus		opción		3	Velocidad en baudios de la línea Modbus (debe coincidir con la del variador de frecuencia). Opciones: 0 2400, 1 4800, 2 9600, 3 19200, 4 38400, 5 115200
$375	Tiempo de espera de recepción de Modbus	milisegundos	entero	50–250	50	El tiempo que se espera una respuesta de Modbus.
$681	Formato serie de Modbus		opción		0	Paridad de la línea Modbus (debe coincidir con la del variador de frecuencia). Opciones: 0 8 bits sin paridad, 1 8 bits con paridad par, 2 8 bits con paridad impar

Motores paso a paso

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$0	Duración del impulso de paso	microsegundos	decimal	≥ 2.0	5.0	Duración de los impulsos de paso, en microsegundos. Los accionamientos necesitan un mínimo (consulte su manual; los accionamientos de lazo cerrado de la MC-1 necesitan 2.5).
$1	Retardo de reposo de los motores	milisegundos	entero	≤ 65535	255	El tiempo que los accionamientos siguen habilitados tras detenerse el movimiento, para que los ejes se asienten antes de cortar la corriente. 255 los mantiene habilitados.
$2	Invertir el impulso de paso		máscara de ejes		0	Salidas de paso que son activas a nivel bajo.
$3	Invertir la dirección de paso		máscara de ejes		5	Ejes cuya salida de dirección está invertida (si no, el eje se mueve en sentido contrario).
$4	Invertir salida(s) de habilitación de los motores		máscara de ejes		7	Salidas de habilitación de los accionamientos que son activas a nivel bajo (la mayoría de los accionamientos). Con una única señal de habilitación compartida solo cuenta X.
$29	Retardo del impulso	microsegundos	decimal	≤ 20	5.0	El tiempo entre fijar las salidas de dirección y el impulso de paso.
$37	Motores que se mantienen habilitados		máscara de ejes		7	Accionamientos que siguen habilitados cuando la máquina está en reposo o en alarma.
$680	Retardo de habilitación de los motores	ms	entero	≤ 500	0	El tiempo entre habilitar los accionamientos y el primer paso.

Variador de frecuencia

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$460	Dirección Modbus del variador del husillo		entero	≤ 255	1	La dirección Modbus del variador de frecuencia.

Eje X

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$100	Resolución de recorrido del eje X	pasos/mm	decimal		800.00000	Pasos por milímetro del eje: pasos por vuelta del motor multiplicados por los micropasos y divididos por el paso del tornillo.
$110	Velocidad máxima del eje X	mm/min	decimal		3000.000	La velocidad más alta del eje, también en rápidos.
$120	Aceleración del eje X	mm/s²	decimal		300.000	La aceleración del eje.
$130	Recorrido máximo del eje X	mm	decimal		202.000	El recorrido del eje desde su posición de referencia, para los límites por software.
$180	Avance de localización del eje X en la búsqueda de referencia	mm/min	decimal		50.0	El avance lento que localiza con exactitud el punto de accionamiento del final de carrera del eje.
$190	Avance de búsqueda del eje X en la búsqueda de referencia	mm/min	decimal		1200.0	El avance rápido con el que se busca el final de carrera del eje.
$1000	Conformador de entrada del eje X		opción		0	Da forma al movimiento del eje para que no excite una resonancia de la máquina a la frecuencia indicada a continuación: ZV es exacto en ella; los demás eliminan una banda más ancha, con menos exactitud. El movimiento queda en reposo, como máximo, 1.5 periodos más tarde. Opciones: 0 Ninguno, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI
$1010	Frecuencia del conformador de entrada del eje X	Hz	decimal	15–500	100.0	La resonancia que elimina el conformador de entrada del eje.
$1020	Relación de amortiguamiento del conformador de entrada del eje X		decimal	≤ 0.5	0.100	La relación de amortiguamiento de esa resonancia.

Eje Y

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$101	Resolución de recorrido del eje Y	pasos/mm	decimal		800.00000	Pasos por milímetro del eje: pasos por vuelta del motor multiplicados por los micropasos y divididos por el paso del tornillo.
$111	Velocidad máxima del eje Y	mm/min	decimal		3000.000	La velocidad más alta del eje, también en rápidos.
$121	Aceleración del eje Y	mm/s²	decimal		300.000	La aceleración del eje.
$131	Recorrido máximo del eje Y	mm	decimal		152.000	El recorrido del eje desde su posición de referencia, para los límites por software.
$181	Avance de localización del eje Y en la búsqueda de referencia	mm/min	decimal		50.0	El avance lento que localiza con exactitud el punto de accionamiento del final de carrera del eje.
$191	Avance de búsqueda del eje Y en la búsqueda de referencia	mm/min	decimal		1200.0	El avance rápido con el que se busca el final de carrera del eje.
$1001	Conformador de entrada del eje Y		opción		0	Da forma al movimiento del eje para que no excite una resonancia de la máquina a la frecuencia indicada a continuación: ZV es exacto en ella; los demás eliminan una banda más ancha, con menos exactitud. El movimiento queda en reposo, como máximo, 1.5 periodos más tarde. Opciones: 0 Ninguno, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI
$1011	Frecuencia del conformador de entrada del eje Y	Hz	decimal	15–500	100.0	La resonancia que elimina el conformador de entrada del eje.
$1021	Relación de amortiguamiento del conformador de entrada del eje Y		decimal	≤ 0.5	0.100	La relación de amortiguamiento de esa resonancia.

Eje Z

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
$102	Resolución de recorrido del eje Z	pasos/mm	decimal		800.00000	Pasos por milímetro del eje: pasos por vuelta del motor multiplicados por los micropasos y divididos por el paso del tornillo.
$112	Velocidad máxima del eje Z	mm/min	decimal		1500.000	La velocidad más alta del eje, también en rápidos.
$122	Aceleración del eje Z	mm/s²	decimal		200.000	La aceleración del eje.
$132	Recorrido máximo del eje Z	mm	decimal		82.000	El recorrido del eje desde su posición de referencia, para los límites por software.
$182	Avance de localización del eje Z en la búsqueda de referencia	mm/min	decimal		50.0	El avance lento que localiza con exactitud el punto de accionamiento del final de carrera del eje.

Ajuste	Nombre	Unidad	Tipo	Rango	MC-1	Descripción
`$192`	Avance de búsqueda del eje Z en la búsqueda de referencia	mm/min	decimal		600.0	El avance rápido con el que se busca el final de carrera del eje.
`$1002`	Conformador de entrada del eje Z		opción		0	Da forma al movimiento del eje para que no excite una resonancia de la máquina a la frecuencia indicada a continuación: ZV es exacto en ella; los demás eliminan una banda más ancha, con menos exactitud. El movimiento queda en reposo, como máximo, 1.5 periodos más tarde. Opciones: 0 Ninguno, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI
`$1012`	Frecuencia del conformador de entrada del eje Z	Hz	decimal	15–500	100.0	La resonancia que elimina el conformador de entrada del eje.
`$1022`	Relación de amortiguamiento del conformador de entrada del eje Z		decimal	≤ 0.5	0.100	La relación de amortiguamiento de esa resonancia.

Códigos de alarma

Las alarmas del controlador. Una alarma detiene la máquina y la bloquea hasta que usted la borra: `$X` o la búsqueda de referencia, según indique la alarma. `$EA` las lista en el propio controlador.

Código	Significado
`ALARM:1`	Límite por hardware: se accionó un final de carrera durante el movimiento. Probablemente se perdió la posición; vuelva a referenciar
`ALARM:2`	Límite por software: el destino del movimiento está fuera del recorrido de la máquina. La posición se conserva; puede desbloquear
`ALARM:3`	Reinicio o parada de emergencia durante el movimiento. Probablemente se perdió la posición; vuelva a referenciar
`ALARM:4`	Palpador: el palpador no estaba en el estado esperado cuando empezó el ciclo de palpado
`ALARM:5`	Palpador: no hubo contacto dentro del recorrido programado
`ALARM:6`	Búsqueda de referencia: el ciclo de búsqueda de referencia se reinició
`ALARM:7`	Búsqueda de referencia: se abrió la puerta de seguridad durante la búsqueda de referencia
`ALARM:8`	Búsqueda de referencia: el retroceso no liberó el final de carrera. Compruebe la distancia de retroceso ($27) y el cableado
`ALARM:9`	Búsqueda de referencia: no se encontró ningún final de carrera dentro de la distancia de búsqueda. Compruebe los ajustes de recorrido y el cableado
`ALARM:10`	Parada de emergencia. Libérela y luego reinicie
`ALARM:11`	Se requiere búsqueda de referencia: ejecute el ciclo de búsqueda de referencia ($H)
`ALARM:12`	Hay un final de carrera accionado. Libérelo antes de continuar
`ALARM:13`	Protección del palpador: el palpador se activó fuera de un ciclo de palpado
`ALARM:14`	El husillo no alcanzó su velocidad a tiempo
`ALARM:15`	Búsqueda de referencia: no se encontró el segundo final de carrera de un eje con escuadrado automático
`ALARM:16`	Falló la autoprueba de encendido
`ALARM:17`	Un accionamiento de motor notifica un fallo
`ALARM:18`	Búsqueda de referencia: los ajustes de la búsqueda de referencia no son válidos
`ALARM:19`	Modbus: sin respuesta o respuesta incorrecta de un dispositivo (p. ej. el variador de frecuencia)
`ALARM:20`	El expansor de E/S no responde
`ALARM:21`	Falló el almacenamiento no volátil (memoria de ajustes)
`ALARM:22`	Se desbordó un búfer

Códigos de error

La respuesta `error:N` a una línea que el controlador rechazó. La línea no hizo nada. `$EE` los lista en el propio controlador.

Código	Significado
`error:1`	Una palabra de código G no tiene letra: cada palabra es una letra seguida de un número
`error:2`	Falta un número o el número no es válido
`error:3`	Comando «$» desconocido o no admitido
`error:4`	Se indicó un valor negativo donde solo se admiten valores positivos
`error:5`	La búsqueda de referencia no está habilitada en los ajustes ($22)
`error:6`	El impulso de paso debe durar al menos 2 microsegundos
`error:7`	No se pudieron leer los ajustes guardados; los ajustes afectados volvieron a sus valores predeterminados
`error:8`	Este comando «$» solo se acepta con la máquina en reposo
`error:9`	No se admite código G durante una alarma o un desplazamiento manual
`error:10`	Los límites por software requieren la búsqueda de referencia: habilite antes la búsqueda de referencia ($22)
`error:11`	La línea era más larga de lo que admite el controlador y no se ejecutó
`error:12`	El valor haría que la frecuencia de pasos fuera superior a la que el controlador puede generar
`error:13`	La puerta de seguridad está abierta
`error:14`	El texto es demasiado largo para guardarse (línea de inicio o información de compilación)
`error:15`	El destino del desplazamiento manual está fuera del recorrido de la máquina; no se realizó el desplazamiento
`error:16`	Un comando de desplazamiento manual necesita «=» y solo puede contener palabras de desplazamiento manual
`error:17`	El modo láser necesita una salida PWM del husillo
`error:18`	El controlador se reinició
`error:19`	El valor debe ser mayor que cero
`error:20`	El bloque contiene un comando de código G no admitido o no válido
`error:21`	Dos comandos del mismo grupo modal en un bloque
`error:22`	No se ha indicado ningún avance
`error:23`	Este comando necesita un número entero
`error:24`	Dos comandos del bloque necesitan ambos las palabras de eje
`error:25`	Una palabra aparece dos veces en el bloque
`error:26`	El comando necesita palabras de eje y el bloque no tiene ninguna
`error:27`	Número de línea no válido
`error:28`	Falta una palabra de valor que el comando necesita
`error:29`	No se admiten G59.1, G59.2 ni G59.3
`error:30`	G53 solo funciona con G0 y G1
`error:31`	El bloque tiene palabras de eje que ningún comando utiliza
`error:32`	Un arco (G2/G3) necesita al menos una palabra de eje en su plano
`error:33`	El destino del movimiento no es válido
`error:34`	El radio del arco no es válido
`error:35`	Un arco (G2/G3) necesita al menos una palabra de distancia al centro (I, J o K) en su plano
`error:36`	El bloque tiene palabras de valor que ningún comando utiliza
`error:37`	G43.1 necesita el eje de longitud de herramienta configurado
`error:38`	El número de herramienta es mayor que el admitido, o esa herramienta no existe
`error:39`	Un valor está fuera de rango
`error:40`	No se permite mientras haya un cambio de herramienta pendiente
`error:41`	El husillo debe estar en marcha para este movimiento (velocidad de corte constante o movimiento sincronizado con el husillo)
`error:42`	El roscado requiere el plano ZX (G18)
`error:43`	El avance supera el máximo
`error:44`	La velocidad del husillo está fuera de rango
`error:45`	Hay un final de carrera accionado: solo se permite la búsqueda de referencia

Código	Significado
`error:46`	Primero referencie la máquina ($H)
`error:47`	No se conoce la herramienta actual: defínala con M61
`error:48`	Palabras de valor en conflicto
`error:49`	Falló la autoprueba de encendido: apague y vuelva a encender el controlador
`error:50`	La parada de emergencia está activa
`error:51`	Un accionamiento de motor notifica un fallo
`error:52`	El valor del ajuste está fuera de rango
`error:53`	Este controlador no tiene ese ajuste
`error:54`	La posición de retracción está por debajo de la profundidad de taladrado
`error:55`	Dos ejes con escuadrado automático no pueden referenciarse a la vez
`error:56`	Este sistema de coordenadas está bloqueado ($486)
`error:57`	Marca inesperada de inicio o fin de programa (%)
`error:58`	El puerto no está disponible
`error:60`	No se pudo montar la tarjeta SD
`error:62`	No se pudo listar el directorio
`error:63`	No existe ese directorio
`error:64`	La tarjeta SD no está montada
`error:65`	No hay ningún sistema de archivos montado (¿falta la tarjeta?)
`error:66`	El sistema de archivos es de solo lectura
`error:71`	Expresión: operador o función desconocidos
`error:72`	Expresión: división por cero
`error:73`	Expresión: argumento fuera de rango
`error:74`	Expresión: argumento no válido
`error:75`	Expresión: error de sintaxis
`error:76`	Expresión: el resultado no es un número
`error:77`	Se requiere autenticación
`error:78`	Acceso denegado
`error:79`	No se permite mientras haya un evento crítico activo
`error:80`	Esta instrucción de palabra O necesita un programa que el controlador pueda volver a leer (desde su sistema de archivos)
`error:81`	Instrucción de palabra O desconocida o mal situada
`error:82`	Demasiados bloques o llamadas de palabra O abiertos a la vez
`error:83`	No queda espacio para la instrucción de palabra O
`error:84`	No se pudo abrir el archivo
`error:85`	No se pudo formatear el sistema de archivos
`error:86`	No se puede usar el puerto
`error:87`	Hay una herramienta en el husillo
`error:88`	No hay ninguna herramienta en el husillo
`error:89`	No se pudo eliminar el archivo
`error:253`	Error definido por el usuario

Esta referencia se genera a partir del programa y solo está en inglés.

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)

Esta referencia se genera a partir del programa y solo está en inglés.

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.

Protocolo

El núcleo propio de Twinwright habla el protocolo serie de Grbl 1.1, con las extensiones que usan los programas de envío Grbl actuales. Un programa de envío escrito para Grbl funciona con él. Los programas de envío que se han comprobado, y sus ajustes, están en Programas de envío de código G.

Conectar

Dónde	Cómo
El gemelo en Studio	Puerto TCP 2323 de este equipo, un programa de envío a la vez.
El MC-1	Su puerto USB nativo, un puerto serie (COM*n* en Windows). La velocidad en baudios no importa; use 115200 donde un programa de envío la pida. El controlador responde en cuanto el programa de envío activa DTR.

Líneas y respuestas

- **Fin de línea:** una línea termina con LF o CR. Una pareja CR LF o LF CR cuenta una sola vez; dos LF son dos líneas.
- **Respuestas:** el controlador responde a cada línea con `ok` o `error:N`, en el orden en que llegaron las líneas. Códigos de error lista todos los N.
- **Longitud de línea:** hasta 256 caracteres. Una línea más larga se responde con `error:11`.
- **Recuento de caracteres:** el búfer de recepción tiene capacidad para 1024 bytes. Un programa de envío puede tener en tránsito esa cantidad de bytes sin respuesta. `Bf:` en el informe de estado indica el espacio libre.
- **Planificador:** 100 bloques. `Bf:` indica también los bloques libres.
- **Líneas que esperan:** en una retención de avance o con la puerta abierta, las líneas esperan en el búfer. Se responden en cuanto el ciclo vuelve a empezar. Un reinicio, `0x19` (parada), `0x03` o `0x85` (cancelar el desplazamiento manual) descarta las líneas que aún esperan.
- **Tras un error:** la siguiente línea de código G no se ejecuta y se responde con el mismo error. Así sigue hasta que llega un comando `$`, una línea vacía o un reinicio.

Comandos en tiempo real

Un comando en tiempo real es un solo byte. El controlador actúa sobre él en cuanto llega, incluso en mitad de una línea. No forma parte de ninguna línea y no recibe `ok`.

Byte	Comando	Qué hace
`0x18` (Ctrl-X)	Reinicio	Lo detiene todo de inmediato y vacía lo que está en cola. Un reinicio con los ejes en movimiento provoca `ALARM:3`: haga la referencia de nuevo. No se acepta mientras la parada de emergencia está activada.
`0x03` (Ctrl-C)	Reinicio, parada de emergencia activada	Como Ctrl-X; también se acepta mientras la parada de emergencia está activada.
`0x19` (Ctrl-Y)	Parada	Los ejes frenan hasta detenerse y se descartan las líneas que siguen en cola. Se conserva la posición y no se genera ninguna alarma.
`ESC`, luego `0x14`	Rearranque	Reinicia el controlador, en las placas que pueden reiniciarse por sí mismas.
`?`	Informe de estado	Un informe de estado.
`0x80` (o `0x05`)	Informe de estado	Un informe de estado, salvo mientras los informes automáticos están activados.
`0x87`	Informe de estado completo	Un informe de estado con todos los campos.
`~`, `0x81`	Inicio de ciclo	Reanuda tras una retención de avance, la puerta, `M0`/`M1` o un cambio de herramienta.
`!`, `0x82`	Retención de avance	Los ejes frenan hasta detenerse y quedan retenidos. Durante un desplazamiento manual (jog), lo cancela.
`0x83`	Estado del analizador	Informa del estado modal, como hace `$G`.
`0x84`	Puerta de seguridad	Actúa como la apertura de la puerta: retiene y, si el estacionamiento está configurado, estaciona.
`0x85`	Cancelar desplazamiento manual	Termina un desplazamiento manual y descarta las líneas de desplazamiento que siguen en cola.
`0x88`	Parada opcional	Activa o desactiva las pausas de `M1`.
`0x89`	Bloque a bloque	Activa o desactiva el bloque a bloque.
`0x8B`	Modo del volante electrónico	Con un volante conectado: cambia su modo. En caso contrario se ignora.
`0x8C`	Informes automáticos	Activa o desactiva los informes de estado automáticos, cuando `$481` fija su intervalo. En caso contrario se ignora.
`0x90`	Ajuste de avance 100 %	
`0x91`, `0x92`	Ajuste de avance +10 %, −10 %	Rango de 10–200 %.
`0x93`, `0x94`	Ajuste de avance +1 %, −1 %	Rango de 10–200 %.
`0x95`, `0x96`, `0x97`, `0x98`	Ajuste de rápidos	100 %, 50 %, 25 %, 5 %.
`0x99`	Ajuste del husillo 100 %	

Byte	Comando	Qué hace
`0x9A`, `0x9B`	Ajuste del husillo +10 %, −10 %	Rango de 10–200 %.
`0x9C`, `0x9D`	Ajuste del husillo +1 %, −1 %	Rango de 10–200 %.
`0x9E`	Parada del husillo	Solo en una retención de avance: detiene el husillo o lo vuelve a poner en marcha.
`0x9F`	Parada de emergencia por software	Se detiene de inmediato y genera la alarma de parada de emergencia.
`0xA0`	Refrigeración, por chorro	Activa o desactiva la refrigeración por chorro.
`0xA1`	Refrigeración, por niebla	Activa o desactiva la refrigeración por niebla.
`0xA3`	Cambio de herramienta terminado	Confirma un cambio de herramienta manual.

El controlador descarta los demás bytes de control y los bytes `0x80–0xBF` que no tienen ningún comando.

- El **texto UTF-8** de los comentarios y de los mensajes se lee como texto. Un byte de `0x80–0xBF` que pertenece a uno de esos caracteres nunca se toma como un comando en tiempo real.
- **`$39=0`** (caracteres heredados en tiempo real desactivados): `?`, `!` y `~` son texto dentro de las líneas `$` y de los comentarios, y comandos en cualquier otro sitio. Con `$39=1` (el valor por defecto) son siempre comandos.

Bienvenida e identidad

Tras el encendido y tras cada reinicio, el controlador dice:

```
Grbl 1.1f ['$' or '$HELP' for help]
```

Los programas de envío toman una línea que empieza por “Grbl” como un reinicio.

`$I` responde con estas líneas y después `ok`:

Línea	Qué dice
`[VER:Twinwright 1.1f.<date>:<text>]`	La versión del protocolo y su fecha de compilación. `<text>` es lo que guardó `$I=<text>`.
`[OPT:…]`	Letras de opciones, los bloques del planificador (100) y el búfer de recepción (1024).
`[NEWOPT:…]`	Nombres de opciones: `ENUMS`, `RT+` o `RT-` (`$39`), `HOME`, `ES`, `MPG`, `LATHE`, `REBOOT`, `TC`, `SS` (sincronización del husillo), `SED`, `RTC`, `SD`, `FS`, `YM`.
`[FIRMWARE:Twinwright]`	El nombre del firmware.
`[SIGNALS:…]`	Las entradas de control que tiene la placa, como letras.
`[NVS STORAGE:…]`	Dónde se guardan los ajustes.
`[DRIVER:…]`, `[DRIVER VERSION:…]`, `[DRIVER OPTIONS:…]`, `[BOARD:…]`	La placa.
`[KERNEL:<version>+<build>]`	La versión y la compilación del controlador: el commit de una versión publicada, o `dev`.
`[KNCAP:…]`	Lo que tiene la placa (más abajo).
`[PLUGIN:…]`	Funciones con el nombre con que las conocen los programas de envío: `MODBUS` y `HUANYANG VFD` para el variador del husillo.

Lo que tiene la placa: KNCAP

`[KNCAP:]` es una línea de pares `key=value`, separados por comas. Las claves solo se añaden al final y nunca cambian de significado. Lea la línea por clave, no por posición. Una clave que falta significa “desconocido”.

Clave	Valores	Significado
`port`	`teensy41`, `stm32h7`, `stm32f4`, `twin`, `host`, `sim`, `unknown`	Qué compilación del controlador es esta.
`pv`	un número	La versión de la interfaz del port del controlador.
`view`	un número	La versión de la vista de estado del controlador.
`spindle`	`vfd`, `pwm`, `none`	El variador del husillo: Modbus, una salida PWM o ninguno.
`encoder`	`0`, `1`	Hay un encoder del husillo montado.
`sd`	`0`, `1`	La placa tiene una ranura para tarjeta SD.
`mpg`	`0`, `2`	Un puerto de volante, indicado por su número de modo.
`aux`	`di/do/ai/ao`	Entradas y salidas auxiliares: entrada digital, salida digital, entrada analógica, salida analógica.

Clave	Valores	Significado
`axes`	letras	Los ejes, por ejemplo `XYZ`.
`probe`	`0`, `1`	Una entrada de palpador. En el MC-1 también la usa el sensor de longitud de herramienta.
`link`	`usb`, `uart`, `tcp`, `none`	La conexión del programa de envío. `usb` es un puerto serie USB que responde en cuanto DTR está activo.

El modo sombra compara `[KERNEL:]` y `[KNCAP:]` entre una máquina real y su gemelo (consulte Máquina real y modo sombra).

Informes de estado

Un informe de estado llega tras `?`, `0x80` o `0x87`. Con los informes automáticos activados (`$481`, `0x8C`), llega también por sí solo. Es una sola línea:

```
<Idle|MPos:0.000,0.000,0.000|Bf:100,1023|FS:0,0|Pn:XYZ>
```

Primero va el estado:

- `Idle`, `Run`, `Jog`, `Home`, `Check` (modo de comprobación), `Sleep`, `Tool` (un cambio de herramienta);
- `Hold:0` (retenido) y `Hold:1` (frenando);
- `Door:0` (cerrada, lista para reanudar), `Door:1` (detenido, aún abierta), `Door:2` (abierta: reteniendo o estacionando) y `Door:3` (cerrada, restableciendo);
- `Alarm`.

Los campos que siguen dependen de `$10` y de la situación:

Campo	Significado
`MPos:` o `WPos:`	Posición de máquina o de pieza (bit 0 de `$10`).
`Bf:`	Bloques libres del planificador, bytes libres del búfer de recepción.
`Ln:`	El número de línea del bloque que se ejecuta.
`FS:`	La velocidad de avance, la velocidad del husillo pedida y la velocidad que indica el variador. Un husillo que solo se enciende y se apaga muestra `F:`. En una retención de avance el avance es 0.
`Pn:`	Entradas activas: los límites por letra de eje, `P` palpador, `D` puerta, `H` retención de avance, `S` inicio de ciclo, `R` reinicio, `E` parada de emergencia.
`WCO:`	El decalaje de origen: cada pocos informes y después de un cambio.
`Ov:`	Los ajustes de avance, de rápidos y del husillo, en %: cada pocos informes y después de un cambio.
`A:`	Accesorios: `S` o `C` husillo en sentido horario o antihorario, `F` refrigeración por chorro, `M` refrigeración por niebla.
`Sl:`	La carga del variador del husillo.
`DTG:`	La distancia por recorrer, hasta que los ejes se detienen (bit 13 de `$10`).
`H:`	Si la referencia está hecha o no.
`T:`	La herramienta, durante un cambio de herramienta.
`TLR:`	Hay una longitud de herramienta de referencia fijada (`$TLR`).
`MPG:`	El volante tiene el control (1) o no (0).
`In:`	El resultado del último `M66`.
`WCS:`, `Sc:`, `D:`	El sistema de coordenadas de pieza, el escalado, el modo de diámetro de torno (informes completos).
`AR`, `AR:`	Informes automáticos desactivados, o activados con su intervalo (informes completos).
`FW:`	El nombre del firmware (informes completos).
`SD:`	Hasta dónde ha llegado un programa que se ejecuta desde el sistema de archivos del controlador.

Otras líneas que envía el controlador

Línea	Cuándo
`ALARM:N`	Una alarma. Códigos de alarma lista todos los N.
`[MSG:…]`	Un mensaje: del propio controlador, o `(MSG,…)`, `(PRINT,…)` o `(DEBUG,…)` de un programa.
`[GC:…]`	El estado modal, tras `$G` o `0x83`. Con el bit 9 de `$10`, una vez después de cada cambio.
`[PRB:x,y,z:1]`	El punto donde se disparó el palpador, y si tocó (1) o no (0).
`[TLO:…]`	La compensación de longitud de herramienta.
`[PARAM:n=value]`	Tras `$#=n`.
`[G54:…]` … `[G59.3:…]`, `[G28:…]`, `[G30:…]`, `[G92:…]`, `[T:…]`	Tras `$#`: los decalajes, las posiciones guardadas, la tabla de herramientas.

Línea	Cuándo
`>line:ok`, `>line:error:N`	Se ejecutó una línea de inicio (`$N0`, `$N1`) al iniciar.

Comandos `$`

`$HELP Commands` los lista en el propio controlador.

Comando	Qué hace
`$`, `$HELP`, `$HELP <group>`	Ayuda; los comandos; un grupo de ajustes con detalles.
`$$`, `$<n>`, `$<n>=<value>`, `$$=<n>`, `$+`	Todos los ajustes; un ajuste; cambiarlo; los detalles de un ajuste; todos los ajustes otra vez.
`$SED=<n>`	Describir un ajuste.
`$#`, `$#=<n>`, `$#=<name>`	Los decalajes, el resultado del palpado, las posiciones guardadas; un parámetro.
`$G`	El estado modal.
`$I`, `$I+`, `$I=<text>`	Qué es el controlador (arriba); todo; guardar un texto con él.
`$N`, `$N0=<line>`, `$N1=<line>`	Mostrar o guardar las líneas de inicio.
`$C`	Modo de comprobación: se comprueba el código G y nada se mueve. Un reinicio lo termina.
`$X`	Desbloquear tras una alarma.
`$H`, `$HX`, `$HY`, `$HZ`	Hacer la referencia de todos los ejes o de uno solo.
`$J=<G-code>`	Desplazamiento manual.
`$B`, `$S`, `$O`	Salto de bloque, bloque a bloque, parada opcional: activar o desactivar.
`$SLP`	Suspensión.
`$TLR`, `$TPW`	Tomar el último palpado como longitud de herramienta de referencia; medir la herramienta nueva en la placa de contacto.
`$RST=$`, `$RST=#`, `$RST=*`	Los ajustes de fábrica de la MC-1 (referenciado y límites activados, sus recorridos); borrar los decalajes; ambos, y las líneas de inicio.
`$EA`, `$EAG`, `$EE`, `$EEG`, `$EG`, `$ES`, `$ESG`, `$ESH`, `$E*`	Listar las alarmas, los códigos de error, los grupos de ajustes y los ajustes, como líneas o como CSV.
`$SPINDLES`, `$SPINDLESH`	Los husillos.
`$PINS`, `$PINSTATE`, `$PORTS`	Los pines de la placa; las entradas y salidas auxiliares; los puertos serie.
`$LIM`, `$HSS`, `$LEV`	Las entradas de límite, los interruptores de referencia, los últimos eventos de entrada.
`$RTC`, `$RTC=<date and time>`	El reloj del controlador (ISO 8601).
`$REBOOT`	Reiniciar el controlador.
`$F`, `$F+`, `$F=<file>`, `$F<=<file>`, `$FR`	Listar los programas del sistema de archivos del controlador, ejecutar uno, enviar uno de vuelta, empezar de nuevo.
`$FD=`, `$FMD=`, `$FRD=`, `$FF=yes`, `$FI`, `$FM`, `$FU`, `$CWD=`, `$PWD`	Eliminar, crear o quitar una carpeta, formatear, información, montar, desmontar, carpeta de trabajo.

Todos los ajustes están en Ajustes.

Formatos de archivo

Borrador (ronda 14): el texto se está escribiendo; las secciones aún pueden crecer.

Programas

Un programa es un archivo de texto de código G en el dialecto de Grbl 1.1, que habla el núcleo propio de Twinwright. Studio lee cualquier archivo de este tipo. Los programas que escribe su CAM tienen este aspecto:

```
(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
```

Parte	Qué es
Cabecera	Comentarios: el título del trabajo; la primera herramienta, el material y los datos de corte principales; dónde está el cero del programa
Modos	`G21 G90 G17 G94 G54`: milímetros, absoluto, el plano XY, avance por minuto, decalaje de origen G54
`G53 G0 Z…`	Un movimiento al extremo superior de Z, en coordenadas de máquina: al principio, en cada cambio de herramienta y al final. La altura es aquella en la que Z queda al hacer la referencia.
`T… M6`	Un cambio de herramienta. El controlador mide la herramienta nueva en el sensor de longitud de herramienta.
`M3 S…`, `G4 P…`	El husillo en marcha y después un tiempo de espera mientras alcanza la velocidad. La bomba de un husillo refrigerado por agua (`M8`) va primero.
Comentarios de elemento	Cada elemento empieza con un comentario que dice qué corta
Movimientos	`G0`, `G1` y arcos `G2`/`G3` con `I` y `J`. El taladrado usa `G81`, o `G83` con descargas, con su plano `R` 1 mm por encima de la parte superior.
Final	`M5`, `M9`, el movimiento al extremo superior, `M30`

Los comentarios van entre paréntesis. Las líneas no llevan números.

Studio guarda cada versión de un programa en la carpeta de programas (Documentos/Twinwright Studio/programs), en una carpeta con el nombre del programa:

Archivo	Qué contiene
`v0001.nc`	El texto del programa, tal como se guardó
`v0001.json`	Su versión, la hora (UTC), quién lo hizo, por qué, la versión de la que procede, su hash y sus líneas
`v0001.ops.json`	En qué se convirtió cada elemento (operación, estrategia, herramienta, líneas, velocidad, avance): lo lee una hoja de amarre

Trabajos y amarres

Un trabajo es la descripción de una pieza a partir de la cual el CAM genera programas. Es un objeto JSON:

Clave	Significado
`name`	El nombre del programa, por ejemplo `bracket.nc`
`title`, `description`	Textos para la cabecera del programa y la hoja de amarre
`material`	El material, por ejemplo `6061-T6`; se entienden los nombres de otras normas
`tool`	La herramienta del husillo, como especificación (`flat 6 n3 L20`)
`tools`	La tabla de herramientas: `{"1": "flat 6 n3 L20", "2": "drill 3.4 118 L20"}`. Si se omite: la de la máquina.
`cutting`	Valores que sustituyen a los elegidos: `rpm`, `feed`, `ap`, `ae`, `finish_ap`; `corner_steps` (true o false: las esquinas interiores de una pared se cortan primero por escalones)
`safe_z`	La altura a la que se hacen los movimientos de traslado, en milímetros sobre el origen de pieza
`zero`	Dónde está el origen de pieza: `front-left-top` (el valor por defecto) o `center-top`
`features`	Los elementos, en el orden en que se cortan. Consulte Crear programas.

Clave	Significado
`setups`	En lugar de `features`: un objeto por cada amarre
`nest`	En lugar de `features`: piezas anidadas en una placa

Un amarre tiene sus propios `features` y dice cómo llega la pieza a él desde el amarre anterior:

Clave	Significado
`turn`	`u`: volteada alrededor de X (el frente y la parte trasera se intercambian); `v`: alrededor de Y (izquierda y derecha)
`fixture`	Se sujeta de otro modo a partir de este amarre (`vise`, `clamps`, `tape` …)
`zero`	El origen de pieza, determinado de nuevo: `front-left-top` o `center-top`. Si se omite tras un volteo o un nuevo utillaje: `front-left-top`; en otro caso, el origen del amarre anterior.
`name`, `tool`, `tools`, `cutting`, `safe_z`	Los propios del amarre, por encima de los del trabajo

```
{"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}]}
```

Cada versión de un trabajo se guarda junto al programa que generó, como `job-v0001.json`: su versión, la hora (UTC), quién lo hizo (`operator` o `ai:<client>`), una nota, la versión del programa que generó y el propio trabajo. Un trabajo descrito en amarres lista además el programa de cada amarre, la versión generada y su hash.

Grabaciones (MCAP)

`--mcap FILE` graba una ejecución en un archivo MCAP (mcap.dev, formato versión 0): un canal para cada tipo de datos, mensajes JSON, cada canal con su esquema JSON, sin fragmentar ni comprimir, y un resumen al final con las estadísticas y un CRC. Las pruebas de Studio vuelven a leer los archivos y los comparan con la especificación de MCAP; las herramientas que leen MCAP, como PlotJuggler y Foxglove, están pensadas para abrirlos, pero todavía no se ha probado.

- `--rec-period MS` fija con qué frecuencia se escriben los canales muestreados (1 ms si no se indica).
- Una grabación no cambia nada en la ejecución: una ejecución grabada y la misma ejecución sin grabar son idénticas bit a bit.
- `--play FILE.mcap` reproduce una grabación en Studio, con una línea de tiempo: reproducir, pausar, buscar, 0.25–10 veces la velocidad. No se simula nada mientras se reproduce.

Tiempo

La hora de registro de cada mensaje es el tiempo virtual del gemelo, en nanosegundos desde el encendido. La mayoría de los canales llevan además `t`, el mismo tiempo en segundos. Un canal muestreado escribe un mensaje solo cuando algo en él ha cambiado.

La cabecera

Un registro de metadatos llamado `mc1sim` dice en qué consistió la ejecución:

Clave	Significado
`machine`	El nombre de la máquina
`drive_model`	El modelo de accionamiento de la ejecución: `L0`, `L1`, `L2`, `L2/3` (tres masas) o `L3`
`f_cpu`	El reloj del controlador, Hz
`nvs`	Un hash de los ajustes guardados del controlador al comenzar la ejecución, 16 dígitos hexadecimales
`sample_period_ms`	El periodo de muestreo
`time`	`virtual, ns from power-on`

Canales

Canal	Escritura	Campo	Tipo	Unidad	Significado
`/machine`	muestreado	`pos`	número[3]	mm	Dónde está cada eje: la mecánica, no la idea que tiene de ello el controlador
		`mpos`	número[3] o null	mm	La posición de máquina del controlador (MPos); null mientras el controlador está apagado
		`lost`	número[3]		Los pasos que ha perdido cada eje
		`feed`	número	mm/min	La velocidad de avance del controlador ahora

Canal	Escritura	Campo	Tipo	Unidad	Significado
		`tip`	número[3]	mm	La punta de la herramienta respecto a la pieza: con el modelo de errores del gemelo activado, desplazada por sus errores; desactivado, igual que `pos`
`/drives`	muestreado, con un modelo de accionamiento desde L1	`ferr_um`	número[3]	µm	El error de seguimiento de cada accionamiento
		`state`	cadena[3]		El estado de cada accionamiento
		`km1`	booleano		El contactor KM1 cerrado
		`bus48`	booleano		La alimentación de 48 V de los accionamientos activada
`/spindle`	muestreado, con un variador del husillo	`f_out`	número	Hz	La frecuencia de salida del variador
		`rpm`	número	rpm	La velocidad del husillo
		`amps`	número	A	La corriente de salida del variador
		`load_pct`	número	%	La carga tal como la indica el variador
		`torque_nm`	número	N·m	El par del motor
		`bus_v`	número	V	La tensión del bus de CC del variador
		`cmd_rpm`	número	rpm	La velocidad que pide el controlador
		`run`, `pump`	booleano		El variador en marcha; la bomba encendida
		`power`	cadena		La alimentación propia del variador: `ready`, `boot` u `off`
		`fault`	cadena		El fallo del variador, tal como lo muestra su pantalla
`/controller`	al cambiar	`state`	cadena		El estado del controlador (`Off` mientras está apagado)
		`alarm`	entero		La alarma en la que está (0: ninguna)
		`homed`	entero		Los ejes con la referencia hecha, un bit cada uno: X 1, Y 2, Z 4
		`limits`	entero		Las entradas de límite que están activas, un bit cada una
		`probe`, `estop`, `motor_fault`, `door`	booleano		Las entradas del palpador, de la parada de emergencia, del fallo de motor y de la puerta (el MC-1 no tiene puerta)
		`on`	booleano		El controlador encendido
`/power`	al cambiar	`mains`, `v24`, `mcu`, `km1`, `bus48`	booleano		La red eléctrica; la alimentación de 24 V; el controlador en marcha; KM1; la alimentación de 48 V
		`vfd_mains`, `vfd`, `pump`	booleano, cadena, booleano		La red del variador del husillo; su estado (`none`, `ready`, `boot`, `off`); la bomba
`/serial/in`	según llega	`text`	cadena		Los bytes que recibió el controlador; los caracteres de control como secuencias de escape `\u`
`/serial/out`	según llega	`line`	cadena		Cada línea que escribió el controlador
`/events`	según ocurren	`text`	cadena		Cada entrada del registro de eventos: marcas, la alimentación, los accionamientos, el variador del husillo, colisiones, cables rotos
`/cut`	muestreado, con un bruto	`cutting`	booleano		Se arranca material; mientras no es así, las cifras siguientes valen 0, salvo `rpm` y `removed_mm3`
		`mrr_mm3_s`	número	mm^3/s	La tasa de arranque de material
		`ap_mm`, `ae_mm`	número	mm	La profundidad axial y el ancho radial del corte
		`fz_mm`	número	mm	El avance por diente
		`rpm`	número	rpm	La velocidad del husillo
		`fx_n`, `fy_n`, `fz_n`	número	N	La fuerza de corte a lo largo de los ejes de la máquina
		`torque_nm`, `power_w`	número	N·m, W	El par y la potencia de corte
		`kc_j_mm3`	número	J/mm^3	La energía de corte específica
		`removed_mm3`	número	mm^3	El material arrancado hasta ahora
`/input`	según llega	`line`	cadena		Las entradas de la sesión, línea a línea tal como las escribe `--record` (consulte Registros de sesión)
`/shadow`	según llega, en modo sombra	`host_s`	número	s	La hora del equipo
		`dir`	cadena		`t` hacia la máquina real, `r` desde ella

Canal	Escritura	Campo	Tipo	Unidad	Significado
		`data`	cadena		Los bytes, cada uno como un carácter U+0000–U+00FF, hasta 1 KB por mensaje
`/ui/state`, `/sound`	para el reproductor				La trama de estado de las vistas (30 por segundo mientras cambia) y la trama del sonido (cada 10 ms de tiempo virtual cuando cambia). Son propias de Studio: sus campos pueden cambiar de una versión a otra.

Trazas de pines (VCD)

`--vcd FILE` escribe los pines del controlador como un archivo VCD (IEEE 1364), que leen GTKWave y PulseView: el ámbito `mcu` contiene los pines del controlador, cada cambio en el tic de reloj en que ocurre (los pines de paso como pulsos de `$0` microsegundos); el ámbito `machine` contiene las líneas ajenas al controlador: las alimentaciones, KM1, el variador del husillo, la bomba, el botón de parada de emergencia, el sensor de longitud de herramienta, los cinco interruptores. La escala de tiempo es 1 ns.

Registros de sesión

`--record FILE` escribe las entradas de una sesión, cada una con su tiempo virtual: lo que enviaron los programas de envío, los comandos en tiempo real, los comandos de las vistas. `--replay FILE` ejecuta la sesión de nuevo, bit a bit, con la misma versión de Studio y los mismos archivos de máquina. `--replay FILE --branch S` la reproduce hasta S segundos y continúa en vivo desde ahí; con `--record`, la rama tiene un registro propio, que parte del registro de origen (`# branch <parent> <ticks>`). Un registro es texto; sus líneas son propias de Studio y pueden cambiar de una versión a otra.

El archivo del mundo

Lo que el gemelo conserva entre sesiones, como hace una máquina real cuando se apaga y se vuelve a encender: dónde están los ejes, el bruto con sus cortes, la herramienta del husillo, la memoria del controlador y el reloj de la placa, y el calor, que se enfría mientras la máquina está apagada. El controlador arranca en frío: se vuelve a hacer la referencia de la máquina.

La ventana guarda el mundo de cada máquina en la carpeta de estado (en Windows `%LOCALAPPDATA%\Twinwright\Studio`): el de la primera máquina en `world`, los de las demás en `machines\<id>\world`. `--world DIR` usa otra carpeta; `--world-reset` parte del estado de fábrica. Las ejecuciones por lotes, las pruebas, las reproducciones, las grabaciones reproducidas y las máquinas en segundo plano de la IA no guardan ningún mundo, salvo que se indique `--world`. Un solo programa a la vez abre un mundo; un segundo parte del estado de fábrica y no guarda nada.

Archivo	Qué contiene
`world.json`	El final de la sesión (más abajo), escrito primero en `world.json.tmp` y luego renombrado
`stock.snap`	Las celdas del bruto, cortadas o no (binario, `TWSTOCK1`)
`eeprom.bin`	La memoria guardada del controlador: sus ajustes, decalajes y datos de herramienta, escrita de inmediato a medida que cambia

`world.json`:

Clave	Significado
`world`	La versión del formato: 1
`saved_at`, `saved_epoch`	Cuándo se guardó: UTC, y segundos desde 1970
`version`	La versión de Studio que lo guardó
`pos`	Dónde estaban los ejes, mm (X, Y, Z)
`tool`	La herramienta del husillo, como especificación (`flat 6 n3 L20`)
`tool_table`	La tabla de herramientas del trabajo: `{"1": "flat 6 n3 L20", ...}`
`tool_auto`	El gemelo hace de operador en un cambio de herramienta: pone la herramienta de la tabla y pulsa inicio de ciclo
`setup`	La sujeción y el bruto, tal como los escribe Guardar amarre en la ventana (null: ninguno)
`cells`	`stock.snap` contiene las celdas del bruto
`thermal`	El calor: `screws` (cada husillo de bolas por encima del bastidor), `frame` (por encima de 20 °C), `ambient` (la sala por encima de 20 °C), `part_heat` y `part_offset` (la pieza por encima del bastidor: lo que aportan los cortes y un decalaje indicado), todo en K; `spindle_um`, el crecimiento del husillo en µm
`rtc`	La lectura del reloj de la placa, s
`probe_cal`	El palpador tal como se calibró por última vez: el diámetro de la bola, su radio efectivo, la lobulación y su fase, el diámetro del anillo patrón, la mayor desviación, cuándo y por quién (solo una vez calibrado)
`odometer`	`travel_mm` y `reversals` de cada eje, `spindle_h` (las horas del husillo) y `on_h` (las horas del programa), acumulados a lo largo de las sesiones

En el siguiente arranque los ejes están donde estaban; vuelve el mismo bruto con sus celdas, y la herramienta y la tabla; el calor se ha enfriado durante el tiempo que la máquina estuvo apagada, cada parte hacia su estado de reposo con su propia constante de tiempo; el reloj de la placa ha seguido avanzando. `--world-off s` indica cuánto tiempo estuvo apagada la máquina (para pruebas; de lo contrario, el tiempo transcurrido desde que se guardó). `--pos`, `--tool`, `--stock`, `--setup` y `--part` indicados al arrancar prevalecen sobre el archivo.

Archivos de licencia

Un archivo de licencia (`.twlic`) es un objeto JSON en UTF-8, firmado con Ed25519. Studio lo comprueba en el equipo, sin red. Dónde están los archivos y cómo se usan: Licencias, puestos y activación.

Este ejemplo procede de los vectores de prueba compartidos. Está firmado con la clave de prueba (`kid` `test`), que las compilaciones publicadas no aceptan:

```
{"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"}
```

Campos

Campo	Tipo	Significado
`v`	entero	La versión del formato: 1. Studio trata un archivo de otra versión como no válido.
`kid`	cadena	Cuál de las claves de Twinwright firmó el archivo
`id`	cadena	El número de la licencia: `TWL-` y 10 letras y dígitos. La renovación lo envía.
`edition`	cadena	`maker` o `pro`
`kind`	cadena	`subscription`, `perpetual` (compra perpetua) o `owner` (propietario del MC-1)
`seats`	entero	El número de puestos
`holder`	cadena	A quién corresponde la licencia: una persona o una organización. Nunca una dirección de correo electrónico.
`issued`	fecha y hora	Cuándo se firmó el archivo, UTC, `YYYY-MM-DDTHH:MM:SSZ`
`valid_until`	fecha y hora, o null	Una suscripción: el final del periodo pagado. Compra perpetua: el final del periodo de reembolso en el primer archivo, null en el permanente.
`renew_after`	fecha y hora, o null	Desde cuándo intenta Studio renovar el archivo
`updates_until`	fecha, o null	Compra perpetua, propietario del MC-1: las versiones publicadas hasta este día tienen las funciones de pago
`fp`	lista de cadenas	Los equipos a los que está vinculada la licencia: los hashes de sus huellas, hasta tres por puesto
`mc1_serial`	cadena, o null	Propietario del MC-1: el número de serie de la máquina. Studio no comprueba la máquina cuando se ejecuta.
`sig`	cadena	La firma Ed25519 de los demás campos, en base64url sin relleno

Un campo que Studio no conoce forma parte de los bytes firmados, y Studio ignora lo que significa.

Los bytes firmados

La firma cubre el archivo sin `sig`, escrito en una forma canónica:

- claves en el orden de los puntos de código Unicode;
- sin espacios en blanco;
- cadenas con las secuencias de escape más cortas de JSON;
- enteros sin punto decimal; ningún otro número, y ninguna clave repetida.

Studio escribe el archivo en esta forma antes de comprobar la firma. Por tanto, los espacios, los saltos de línea y el orden de las claves del archivo no importan. Un valor cambiado hace fallar la comprobación.

El hash de la huella

El hash son los primeros 32 dígitos hexadecimales de SHA-256 sobre los bytes `twinwright-fp-v1`, un byte cero y el ID del equipo:

Sistema	El ID del equipo
Windows	El valor `MachineGuid` en `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography`
Linux	`/etc/machine-id`

Si no hay ninguno de los dos, Studio crea un ID aleatorio y lo guarda en la carpeta de licencias (`machine.id`); al borrarlo, el equipo pasa a ser otro distinto. El ID en sí nunca sale del equipo.

Por ejemplo, el ID `8b1c3a0e-5f2d-4c7a-9e61-2d0f4b7a1c55` da el hash `0a2d31bb82176516415efa7b3a340562`.

El código de solicitud

El código de solicitud para la activación sin conexión es el JSON canónico en UTF-8 `{"fp": <the hash>, "version": <Studio's version>}`, en base64url sin relleno `=`. Por ejemplo, el hash `0123456789abcdef0123456789abcdef` y la versión 1.0.0 dan:

```
eyJmcCI6IjAxMjM0NTY3ODlhYmNkZWYwMTIzNDU2Nzg5YWJjZGVmIiwidmVyc2lvbiI6IjEuMC4wIn0
```

`TwinwrightStudio.exe --licence-request` imprime el código de solicitud de este equipo.

Archivos de prueba

Una prueba es un archivo de licencia como los demás, firmado por el servidor de licencias de Twinwright: `"kind": "trial"`, el hash de este equipo en `fp` y `valid_until` 30 días después de empezar la prueba. Nunca se renueva. Un archivo de prueba modificado a mano deja de ser válido.

El archivo de la máquina

Todas las claves del archivo de una máquina, con los valores del MC-1: Esquema del perfil de máquina.

Programas de envío con el núcleo propio de Twinwright

Esta página es para quienes quieran manejar el MC-1 desde un programa de envío de código G propio (bCNC, gSender o Universal G-code Sender) en lugar de hacerlo desde Studio. El controlador del MC-1 es el núcleo propio de Twinwright, compatible con el protocolo serie de Grbl 1.1 y con las extensiones que usan los programas de envío actuales, de modo que estos programas de envío funcionan con él, con los pocos ajustes y salvedades que se indican a continuación.

Todo lo que aparece aquí se probó en el gemelo digital del MC-1, que ejecuta el mismo código de controlador que la máquina. La comprobación en una máquina real está aún por hacer. Versiones probadas: bCNC 0.9.16, gSender 1.6.4, UGS Platform 2.1.27.

Conectar

Dónde	Cómo
El gemelo en la ventana de Studio	Studio admite un programa de envío a la vez en `localhost:2323` (solo en este equipo; inicie Studio con `--bind any` para acceder a él desde la red). bCNC: puerto `socket://localhost:2323`. gSender: Ethernet, dirección `127.0.0.1`, puerto `2323`. UGS: `tcp://localhost:2323`.
El propio MC-1	USB nativo: el controlador aparece como un puerto serie (COM*n* en Windows). Elija ese puerto. La velocidad en baudios no importa en USB nativo, pero algunos programas de envío piden una: 115200 funciona.

Lo que el controlador dice de sí mismo:

- Su línea de bienvenida es `Grbl 1.1f ['$' or '$HELP' for help]`, como la de un controlador Grbl clásico. Los programas de envío buscan una línea de bienvenida que empiece por "Grbl" para darse cuenta de que el controlador se reinició (consulte la sección 3).
- Su nombre figura en la respuesta a `$I`: `[VER:Twinwright 1.1f.…]` y `[FIRMWARE:Twinwright]`. Los programas de envío que muestran el firmware muestran "Twinwright".

Elegir el firmware en el programa de envío

Programa de envío	Ajuste	Motivo
bCNC 0.9.16	Controller: **GRBL1**	bCNC no lee el nombre del firmware. Probado con GRBL1.
gSender 1.6.4	Default firmware: la opción distinta de **Grbl**	gSender reconoce un firmware por su nombre en `$I`. Nuestro controlador da su propio nombre, así que gSender usa el firmware que usted haya fijado como predeterminado. Probado con esa opción.
UGS 2.1.27	Firmware: **GRBL**	Probado: ajustes, programas, alarmas y referencia tras un reinicio, sin un Soft Reset adicional (sección 3). La otra opción de firmware de UGS también funciona, pero necesita un Soft Reset más tras un reinicio; el desplazamiento manual (jog) se probó solo con esa opción.

Tras una alarma o un reinicio

- **Parada de emergencia.** La parada de emergencia del MC-1 abre el contactor de potencia. Suéltela, pulse el botón Start de la máquina, reinicie después desde el programa de envío y haga la referencia. Hasta que se pulsa Start, la alarma (ALARM:10) no se borra.
- **ALARM:11 (se requiere hacer la referencia).** Haga la referencia (`$H`). El desbloqueo con `$X` se rechaza (error:46).
- **ALARM:2 (límite por software).** El programa pidió un movimiento fuera del recorrido. La referencia deja Z en el extremo superior de su recorrido (Z de máquina 0), de modo que cualquier movimiento por encima de ese punto, como `G53 G0 Z5`, provoca ALARM:2. Reinicie y después haga la referencia o desbloquee.
- **bCNC** se recupera tras cualquier reinicio: ve la línea de bienvenida "Grbl" y olvida las líneas que aún estaban en camino, así que `$H` se envía.
- **bCNC tras Stop.** El Stop de bCNC es una retención de avance seguida de un reinicio. Después, bCNC devuelve al controlador los modos que tenía antes (sistema de coordenadas, unidades, compensación de longitud de herramienta) en una o dos líneas, y en algunos casos el controlador las rechaza, como hacen otros controladores Grbl: con una compensación de longitud de herramienta (`G43.1`) activa, la línea recibe error:24 (bCNC envía un `G43.1Z...` propio tras su primer Stop, de modo que a partir del segundo Stop ocurre siempre); bajo `G80`, `G43.1Z...` recibe error:31; con el escalado (`G51`) activo, `G51:X` recibe error:1. Tras un error, cada línea de código G que sigue recibe el mismo error hasta que llega una línea vacía, un comando `$` o un reinicio. Por tanto, tras un Stop, compruebe el sistema de coordenadas, las unidades y la compensación de longitud de herramienta con `$G` y `$#` antes de hacer un desplazamiento manual o de continuar. Si son incorrectos, vuelva a fijar el origen de pieza y vuelva a medir la longitud de la herramienta (probado en el gemelo, 2026-10-06).
- **UGS con su otra opción de firmware.** Tras un reinicio que UGS no envió él mismo, por ejemplo uno desde la ventana de Studio o un reinicio del propio controlador, pulse **Soft Reset** una vez más en UGS antes de Home. Hasta entonces Home no hace nada. Con esa opción UGS solo reconoce como reinicio otra línea de bienvenida. Un reinicio desde el propio botón Soft Reset de UGS funciona bien. En el propio MC-1 estos reinicios son raros: no tiene entrada de reinicio y solo un host USB. Con el firmware configurado como "GRBL" esto no ocurre: tras un reinicio así UGS muestra CONNECTING y Home funciona al instante (probado en el gemelo, 2026-10-06).

Cambios de herramienta (M6)

Ninguno de los tres programas de envío, con sus ajustes por defecto, permite que el MC-1 cambie la herramienta y mida la nueva:

Programa de envío	Por defecto	Qué hacer
bCNC	Se salta todas las líneas M6: los cortes de la herramienta siguiente se hacen con la herramienta que aún está en el husillo.	Configure su cambio de herramienta en **Manual tool change (TLO)** y mida en el sensor de longitud de herramienta del MC-1 (sección 5).

Programa de envío	Por defecto	Qué hacer
gSender	Cambio de herramienta "Ignore": M6 se envía como comentario y el programa no se detiene.	Configure su cambio de herramienta en **Fixed Tool Sensor** en el sensor de longitud de herramienta del MC-1 (sección 5).
UGS	Gestiona M6 por sí mismo: retrae, detiene el husillo y le pide que cambie la herramienta, pero nunca la mide. Una herramienta más larga corta entonces demasiado profundo y, en una máquina real, puede chocar con la pieza.	Ejecute una herramienta por programa y fije la longitud de la herramienta o el origen de pieza antes de cada una, o use Studio para los programas con cambios de herramienta.

Más sobre UGS y M6:

- También se detiene en un "M6" dentro de un comentario, como el `( M6 T1 )` de FreeCAD.
- Un segundo M6 que llega justo después de un cambio de herramienta puede colarse hasta el controlador. El controlador espera entonces en el estado Tool una confirmación que UGS nunca envía, y ningún botón de la ventana de UGS funciona ya. En el MC-1 solo la parada de emergencia lo termina: púlsela, suéltela, pulse Start, y después Soft Reset y Home en UGS.
- Si el controlador se reinicia durante un trabajo de UGS antes de que UGS haya llegado a un M6, UGS puede dar el trabajo por terminado, pero su diálogo de cambio de herramienta sigue apareciendo. Cerrarlo puede hacer que todo el programa vuelva a empezar desde el principio. Tras un reinicio a mitad de un trabajo, cierre el archivo en UGS antes de continuar.

Medir la longitud de herramienta en el sensor de longitud de herramienta del MC-1

El sensor de longitud de herramienta está en la parte trasera izquierda de la mesa; `G59.3` se guarda justo encima de él.

- **bCNC, Manual tool change (TLO).** Posición de cambio de herramienta encima del sensor de longitud de herramienta. Posición de palpado `G59.3`. Distancia de palpado 50 mm, búsqueda 200 mm/min, lento 25 mm/min (`$342`–`$344` del MC-1), comando G38.2. Mida la primera herramienta con "Calibrate" antes de ejecutar el programa. Después bCNC fija `G43.1` para cada herramienta nueva.
- **gSender, Fixed Tool Sensor.** Ubicación del sensor `G59.3`. Mida la herramienta inicial en el asistente. Después gSender desplaza la Z de pieza para cada herramienta nueva (`G10 L20`) en lugar de usar una compensación de longitud de herramienta. Toma la longitud de donde se detuvo el eje tras el contacto, no del punto de disparo que indica el controlador, por lo que cada cambio puede quedar algo desviado. En el gemelo, ejecute los cambios de herramienta a velocidad 1×: gSender lee esa posición tras una pausa corta fija, y un gemelo que funciona más rápido puede darle una posición anterior a la parada.
- **Empiece al menos 0.1 mm por debajo del extremo superior de Z** (Z de máquina −0.1 o menos) cuando comience el cambio de herramienta. Ambos programas de envío vuelven a esa altura tras medir. Si se empieza justo en el extremo superior, la pequeña dispersión de una medición puede situar ese retorno por encima del límite por software, y el trabajo se detiene con ALARM:2 antes de cortar.

Fijar el origen de pieza con un palpador

- **UGS** (su módulo de palpado) calcula el origen de pieza a partir de los puntos de disparo que indica el controlador (`[PRB:]`). Esta es la forma correcta.
- **gSender** (palpador 3D) y **bCNC** (su página de palpado y después sus botones Z=0 / X / Y) fijan el origen de pieza donde se detuvieron los ejes tras el contacto. La máquina siempre sigue un poco más allá del disparo antes de detenerse, y cuanto más rápido es el contacto final, más lejos llega. Use un contacto final lento, o fije el origen de pieza con los ciclos de palpado de Studio, que usan los puntos de disparo.
- **El centro de agujero de UGS** (el Hole Center de su módulo de palpado): desde donde usted deja el palpador, palpa -X, +X, -Y y +Y, cada uno a la mitad del diámetro del agujero que usted introduce, y fija el centro a medio camino entre los puntos de disparo de los contactos lentos. El alcance depende de ese diámetro: el punto de partida puede estar como máximo a (diámetro introducido - diámetro real del agujero) / 2 + el radio de la bola respecto del centro. Con el propio diámetro del agujero introducido, eso es solo el radio de la bola (1 mm para una bola de 2 mm); introduzca uno unos mm mayor (24 a 26 para un agujero de 20 mm) para dejar margen al colocar el palpador a ojo.
- **Cuando el centro de agujero de UGS no llega a la pared** (hallazgo 192, comportamiento propio de UGS): ese palpado termina en ALARM:5 y UGS se detiene. Pero antes de palpar fijó X0 Y0 de G54 en el punto de partida y, como el ciclo no terminó, G54 se queda ahí. Tras desbloquear, vuelva a fijar el origen de pieza antes de cortar. Su ciclo de palpado también queda a medias: desconecte y vuelva a conectar antes de palpar de nuevo. Además toma por un contacto un punto que el controlador indica como no tocado (`[PRB:...:0]`).
- Calibre primero el palpador (ciclos de palpado de Studio: un anillo patrón). Un palpador sin calibrar mide cada agujero más grande y cada saliente más pequeño en el doble de su recorrido previo.

Otras cosas que conviene saber

- **bCNC por la red.** En Windows, bCNC lee una conexión TCP solo cada 2 segundos aproximadamente, por lo que su pantalla puede mostrar datos de hace 2 segundos. "Idle" en bCNC no significa que la máquina se haya detenido. Una conexión USB no tiene este retraso.
- **gSender, M0.** Un M0 dentro de las 10 primeras líneas de un programa no hace pausa (gSender lo convierte en un comentario). M1 siempre hace pausa.
- **gSender, desplazamiento manual continuo.** Mantener pulsado un botón de desplazamiento manual se rechaza en el MC-1 (error:15): gSender pide un desplazamiento largo más allá del recorrido. Haga el desplazamiento por pasos. En algunos controladores Grbl, `$40=1` (limitar los comandos de desplazamiento manual) termina los desplazamientos largos en los límites por software en lugar de rechazarlos; el controlador del MC-1 todavía no actúa según ese ajuste y sigue rechazándolos (probado con gSender en el gemelo, 2026-10-06).
- **UGS, líneas largas.** UGS nunca envía una línea de más de 127 caracteres: el trabajo se detiene sin ningún mensaje. Mantenga las líneas más cortas (UGS elimina antes los comentarios).
- **UGS, M30.** UGS elimina M30. Un programa que cuenta con M30 para detener el husillo y el refrigerante los deja en marcha: termine los programas con M5 y M9 antes de M30.
- **UGS, comentarios.** Si una línea tiene dos comentarios, UGS elimina también el código que hay entre ellos.
- **Desconectar durante un trabajo.** El controlador sigue ejecutando las líneas que ya tiene. gSender y UGS no envían ninguna parada al desconectar, así que la máquina sigue funcionando, con el husillo girando. Detenga el trabajo antes de desconectar. bCNC hace una retención y un reinicio antes de cerrar la conexión.

- **Comentarios con caracteres no ASCII** (como los caracteres chinos, "Ø", "–"): los tres programas de envío eliminan los comentarios antes de enviar, de modo que estos caracteres nunca llegan al controlador. El propio controlador lee como texto el texto UTF-8 de los comentarios y de `(MSG,...)`, nunca como un comando en tiempo real.