Twinwright Studio 1.2.1

Twinwright Studio manual

2026-10-16

Sommaire

User manual

1. Premiers pas
2. La fenêtre et les espaces de travail
3. Machines et mise sous tension
4. Préparer un travail
5. Palpage et recherche de l'origine pièce
6. Créer des programmes (FAO)
7. Programmes et éditeur
8. Exécuter un programme et le reprendre
9. Analyse et diagnostic
10. Exécuter un programme dans le jumeau
11. Contrôle, rapports et fiches de bridage
12. Connecter une machine réelle
13. L'assistant IA
14. Langue et unités
15. Clavier et accessibilité
16. Dépannage

Operations manual

1. Configuration requise
2. Installer et désinstaller
3. Licences, postes et activation
4. Vos données et sauvegardes
5. Mettre à jour Studio
6. Confidentialité et ce que Studio envoie
7. Laboratoires, salles de cours et ordinateurs partagés

Technical manual

1. Les modèles du jumeau et leurs limites
2. Le noyau propre à Twinwright
3. G-code et M-code
4. Stratégies et paramètres de FAO
5. Ajustements, faux-rond et méthodes de contrôle
6. Profils de machine
7. Matériel de la MC-1

Reference

1. MCP tool reference
2. Référence des réglages
3. Codes d'alarme
4. Codes d'erreur
5. Command line
6. Machine profile schema
7. Protocole
8. Formats de fichiers
9. Logiciels d'envoi avec le noyau propre à Twinwright

User manual

Premiers pas

Twinwright Studio fait tourner un jumeau numérique de votre machine CNC. Vous préparez la pièce, vous créez ou ouvrez un programme et vous l'exécutez dans le jumeau avant que la machine réelle ne coupe. Ce chapitre vous mène de l'installation à une première pièce en une dizaine de minutes.

Installation

Studio nécessite Windows 10 ou 11 en 64 bits (voir Configuration requise).

1. Lancez `TwinwrightStudio-<version>-setup.exe`. Vous n'avez pas besoin des droits d'administrateur : Studio est installé pour vous.
2. Lisez le contrat de licence et cliquez sur **I Agree**.
3. Gardez le dossier, ou choisissez-en un autre.
4. Sur la dernière page, laissez **Run Twinwright Studio** coché et cliquez sur **Finish**.

Dans le menu Démarrer, Studio s'appelle **Twinwright Studio**. L'installation sur de nombreux ordinateurs et l'emplacement des fichiers : Installer et désinstaller.

Le premier démarrage

Studio s'ouvre dans l'espace de travail **Exécution**, avec la machine **MC-1 · Mon établi** sous tension. Rien n'est demandé.

La barre machine affiche **ALARM:11** et **Étape suivante : Origine**. C'est normal : comme une machine réelle, le jumeau ne sait pas où se trouvent ses axes tant qu'il n'a pas fait sa prise d'origine.

Ce que la fenêtre affiche

Une nouvelle installation affiche tout ce qu'utilise un ingénieur : les six espaces de travail, les courbes et les colorations. Si vous ne faites qu'exécuter des programmes, ou si vous enseignez, choisissez un niveau qui en affiche moins : **Outils > Niveau d'interface** (voir Niveaux d'utilisateur).

Langue et unités

Studio démarre dans la langue de votre système lorsqu'il dispose de cette langue (anglais, chinois, français, allemand, japonais ou espagnol), et sinon en anglais. Les unités et la présentation des nombres suivent votre région.

Pour les modifier : **Outils > Language / 语言** et **Outils > Unités**. Voir Langue et unités.

Éditions et essai

L'édition que vous avez s'affiche dans la barre supérieure, par exemple **Learning**. Cliquez dessus, ou utilisez **Aide > Licence…**, pour la voir.

Édition	Ajoute
Learning	Poches, contours, perçage, gravure et finition 3D en balayage parallèle ; tout le jumeau, l'éditeur et l'IA
Maker	Évidement adaptatif, toutes les stratégies de finition 3D, reprise des restes, deux faces et plusieurs phases
Pro	Imbrication, optimiseur de programme, rapports (le rapport de contrôle, la fiche de bridage), contrôle au palpeur par rapport à la conception

Chaque édition comprend tout ce qu'offrent les éditions situées au-dessus d'elle dans le tableau. Une fonction d'une édition supérieure s'affiche avec le nom de son édition. Cliquez dessus pour lire ce qu'elle fait, et pour commencer un essai ou ouvrir la licence.

Quand un essai ou une licence prend fin, Studio revient à l'édition Learning. Vos projets, vos programmes et le G-code que vous avez créé continuent de fonctionner.

Commencer un essai

Vous pouvez essayer l'édition Maker et l'édition Pro pendant 30 jours chacune, une fois par ordinateur.

1. Cliquez sur une fonction de cette édition, par exemple **Optimiser** dans l'espace de travail **Programme**.
2. Cliquez sur **Essayer 30 jours…**
3. Cliquez sur **Commencer l'essai**.

Studio demande l'essai une seule fois au serveur de licences de Twinwright, en envoyant l'édition, une empreinte hachée de cet ordinateur et la version de Studio. À la fin de l'essai, une notification dans l'onglet **Messages** l'indique.

Activer une licence

Ouvrez **Aide > Licence…**.

Pour activer en ligne :

1. Saisissez le code d'activation (`TW-XXXXX-XXXXX-XXXXX-XXXXX`). Majuscules, minuscules, espaces et tirets n'ont pas d'importance.
2. Cliquez sur **Activer**. Résultat : **Activation réussie.**

Pour activer un ordinateur sans accès à Internet :

1. Sous **Activer hors ligne…**, copiez le code de demande de cet ordinateur.
2. Sur un ordinateur avec accès à Internet, ouvrez la page des licences, collez le code de demande et téléchargez le fichier de licence (`.twlic`).
3. Rapportez le fichier et cliquez sur **Importer un fichier de licence…**

Renouveler avant l'échéance renouvelle la licence en ligne avant qu'elle ne prenne fin. Cette option n'envoie que le numéro de licence, une empreinte hachée de cet ordinateur et la version de Studio.

Postes, transfert sur un autre ordinateur et salles de cours : Licences.

Une première pièce en dix minutes

La première pièce la plus rapide à faire est l'une des pièces d'exemple : Studio met en place le brut, les outils et l'origine pièce pour vous.

Ouvrir la machine

La machine est déjà ouverte : **MC-1 · Mon établi**, dans l'espace de travail **Exécution**. Les autres machines se trouvent dans **Fichier > Bibliothèque de machines…**.

Mettre sous tension et faire la prise d'origine

Appuyez sur **Origine** (F3), ou sur la touche que la barre machine propose.

Résultat : les trois axes trouvent leur origine l'un après l'autre, en 20 s environ. La barre machine affiche **Aucune alarme**.

Si la barre machine affiche **Arrêt d'urgence**, relâchez-le d'abord (F1), puis appuyez sur **Mise sous tension** (F2) et sur **Reset** (F12).

Ouvrir un programme d'exemple

1. Choisissez **Fichier > Pièces d'exemple…**
2. Choisissez **Plaque de moteur NEMA 17** (e01_motor_mount) : un alésage ajusté, des lamages et des rainures ; la pièce est détachée sur des attaches.
   Résultat : Studio place le brut, charge les outils et le programme, et définit l'origine pièce sur le brut. La console indique quand l'exemple est prêt.

Exécuter le programme dans le jumeau

1. Appuyez sur **Départ cycle** (F5).
2. Pour suivre l'exécution plus vite, choisissez **Simulation > Vitesse > ×10 temps réel**.

La vue 3D montre l'outil en train de couper le brut. **Arrêt d'avance** (F6) met le programme en pause, **Départ cycle** le reprend.

Examiner le résultat

- Faites glisser dans la vue 3D pour la faire tourner ; la molette zoome.
- **Affichage > Coloration > Pièce : écart par rapport au modèle** colore la matière restante à usiner et les surcoupes éventuelles.
- **Analyse** (Alt+5) montre quelles lignes ont pris du temps et les estimations du jumeau pour l'effort de coupe et le broutement.

Mettre en place le brut

Pour votre propre pièce, préparez la machine dans **Préparation** (Alt+1) au lieu de charger un exemple :

1. Allez à l'étape **Bridage**.
2. Sous **Pièce et bridage**, choisissez la **Forme**, la **Matière** et le **Montage**, et saisissez les dimensions.
3. Cliquez sur **Placer**.

Voir Préparer un travail.

Mesurer l'outil

1. Allez à l'étape **Outil**, choisissez l'outil et cliquez sur **Changer d'outil**.
2. Cliquez sur **Mesurer l'outil de référence**.

La machine doit être référencée et la broche arrêtée.

Définir l'origine pièce

1. Allez à l'étape **Zéro** et amenez l'outil, par déplacement manuel, au point que vous voulez prendre pour zéro.
2. Cliquez sur **Tout à zéro**, ou mettez les axes à zéro un par un.

Avec un palpeur, la section Palpeur trouve le zéro pour vous (voir Palpage).

Ouvrez ensuite votre programme (**Fichier > Ouvrir un programme…**) et exécutez-le comme ci-dessus.

Pour aller plus loin

- La fenêtre et les espaces de travail : où se trouve quoi.

- Créer des programmes : la FAO dans Studio.
- Exécuter un programme dans le jumeau : ce que le jumeau vérifie et estime.
- Exécuter un programme et le reprendre : exécutions, alarmes et reprise.

Un **?** sur un panneau ouvre ce manuel au chapitre de ce panneau. **Aide > Documentation** l'ouvre au début.

La fenêtre et les espaces de travail

La fenêtre de Studio comporte en haut les menus et une barre supérieure, au milieu six espaces de travail et en bas la barre machine. Chaque espace de travail sert à une tâche principale. La barre machine montre l'état de la machine et l'étape suivante où que vous soyez.

Les éléments de la fenêtre

De haut en bas :

- la **ligne de titre** : les menus à gauche, le nom de la fenêtre au milieu, les boutons de la fenêtre à droite ;
- la **barre supérieure** : la machine, les onglets des espaces de travail, l'édition, l'état de l'IA, le niveau d'utilisateur et **?** ;
- l'**espace de travail** : la vue 3D et les panneaux de l'espace de travail que vous avez choisi ;
- le **panneau inférieur** sous la vue 3D, dans les espaces de travail qui en ont un ;
- la **barre machine**, la même dans tous les espaces de travail.

Studio n'a pas de barre d'état : la barre machine en tient lieu.

La ligne de titre et les menus

Les menus sont **Fichier**, **Édition**, **Affichage**, **Machine**, **Programme**, **Simulation**, **IA**, **Outils** et **Aide**.

- Appuyez sur Alt et sur la lettre d'un menu pour l'ouvrir : Alt+F ouvre **Fichier**.
- Appuyez sur la touche Alt seule pour mettre en évidence les titres des menus, puis utilisez les touches fléchées, Entrée et Échap.

Tant qu'un menu est ouvert, aucune touche n'atteint la machine. Un menu ne montre que les éléments de votre niveau d'utilisateur (voir Niveaux d'utilisateur).

La barre supérieure

De gauche à droite :

- le logo : il ouvre **Fichier > Bibliothèque de machines…** ;
- la machine : son modèle, son nom et son état. Cliquez dessus, ou appuyez sur Ctrl+Maj+M, pour changer de machine : tapez pour filtrer, choisissez avec ↑ ↓ et Entrée ;
- les onglets des espaces de travail (voir Les six espaces de travail) ;
- l'édition, par exemple **Learning** : cliquez dessus pour ouvrir **Aide > Licence…** ;
- l'état de l'IA : **IA désactivée**, **IA activée**, **IA connectée** ou **L'IA attend votre accord**. Cliquez dessus pour accéder aux paramètres de l'IA (voir L'assistant IA) ;
- le niveau d'utilisateur : cliquez dessus pour en choisir un autre ;
- **?** : la liste des raccourcis clavier.

La barre machine

La barre machine longe le bas de la fenêtre. De gauche à droite :

1. **L'état** : **Repos**, **Exécution**, **Jog**, **Prise d'origine**, **Suspendu**, **Porte**, **Changement d'outil**, **Mode vérification**, **Veille**, **Alarme**, **Arrêt d'urgence** ou **Hors tension**.
2. **Le programme** : son nom, la ligne en cours, la section en cours, une barre de progression, le temps écoulé et le temps restant. Sans programme, la barre affiche **Aucun programme chargé**, si la machine est référencée, l'outil et le décalage d'origine.
3. **L'étape suivante** : **Aucune alarme** en vert, ou **Étape suivante :** et la touche qui l'exécute. En cas d'alarme, la barre affiche l'alarme, l'étape suivante et **Toutes les étapes**, qui liste les étapes du retour à l'état de marche (voir Alarmes et conduite à tenir).
4. **L'horloge** : le temps de la simulation, sa vitesse et la vitesse réelle qu'elle atteint. La touche en forme de sablier met la simulation en pause.
5. **Les touches** : **Bloc à bloc**, **Départ cycle** (F5), **Arrêt d'avance** (F6) et **Arrêt** (F7). La touche que Studio suggère est pleine ; une touche inutilisable pour l'instant est grisée.
6. **Arrêt d'urgence** (F1), la touche ronde et rouge à l'extrémité droite.

Dans une fenêtre étroite, la barre supprime d'abord le temps de l'horloge, puis la section et les temps du programme (ils passent dans l'info-bulle), puis les vitesses de l'horloge, puis les mots de l'étape suivante.

La vue 3D

La vue 3D montre la machine, la pièce et l'outil tels qu'ils sont à cet instant.

- Faites glisser avec le bouton gauche pour faire tourner la vue, avec le bouton droit ou le bouton central pour la déplacer. La molette zoome vers le pointeur.
- Les touches en haut à gauche choisissent une vue : isométrique, de face, de droite ou de dessus. **Suivre l'outil** garde l'outil au milieu. **Affichage** active et désactive la trace de l'outil et le trajet du programme.
- **Gros plan sur l'outil** montre l'outil à la coupe dans une petite image en haut à droite. Il apparaît dans l'espace de travail **Exécution** pendant qu'un programme s'exécute.
- Les pastilles en bas nomment la pièce et l'outil.
- Cliquez sur la pièce ou sur la trace : la barre sous la vue vous dit quelle ligne du programme a produit ce point, et le listing se place sur cette ligne.

Les mêmes choix se trouvent dans **Affichage > Vue 3D**. **Affichage > Coloration** colore la trace et la pièce selon l'avance, l'effort de coupe, l'épaisseur de copeau, l'indice de broutement ou la ligne du programme (à partir du niveau Ingénieur).

Le panneau inférieur

Le panneau inférieur se trouve sous la vue 3D. Ses onglets dépendent de l'espace de travail :

Espace de travail	Onglets
Exécution	**Programme**, **Courbes**, **Console**, **Messages**, **Activité IA**
Préparation	**Résultats du contrôle** (dès qu'une pièce a été mesurée), **Table d'outils**
FAO	**Résultats de vérification**, **Activité IA**, **Versions**, **Aperçu du programme**
Programme	**Console**

Analyse, **Diagnostic** et le mode atelier n'ont pas de panneau inférieur.

- **Programme** : le listing du programme. **Suivre la ligne** garde en vue la ligne en cours d'exécution ; les barres de temps montrent le temps de chaque ligne dans cette exécution ou dans la dernière.
- **Courbes** : les relevés de l'exécution au fil du temps.
- **Console** : ce que Studio et le contrôleur ont dit. Tapez une ligne de G-code ou une commande `$` dans la ligne du bas et appuyez sur Entrée pour l'envoyer ; ↑ et ↓ rappellent les lignes précédentes.
- **Messages** : les avertissements et les constats, regroupés selon leur origine. Un nombre rouge sur l'onglet compte les avertissements.
- **Table d'outils** : les outils de la machine.
- **Résultats du contrôle** : la dernière pièce mesurée avec le palpeur, contrôle par contrôle – sa cote nominale, sa tolérance, la mesure, l'écart et l'estimation du jumeau (voir Contrôle).
- **Activité IA** : ce que des clients IA ont fait sur les machines de cet ordinateur.

Appuyez sur Ctrl+J, ou cliquez sur la flèche à l'extrémité droite des onglets, pour replier le panneau inférieur sur sa rangée d'onglets.

Les six espaces de travail

Changez d'espace de travail avec les onglets de la barre supérieure, avec Alt+1 à Alt+6, avec Ctrl+Tab et Ctrl+Maj+Tab, ou depuis le menu **Affichage**. Appuyer sur **Départ cycle** dans **FAO** ou **Programme** vous amène à **Exécution**. Pendant qu'un programme s'exécute, l'onglet **Exécution** porte un point vert.

Préparation

Préparation (Alt+1) prépare la machine et la pièce, une étape à la fois : **Tension**, **Origine**, **Bridage**, **Outil**, **Zéro**, **Programme** et **Usinage**. Le panneau de droite suit l'étape suivante jusqu'à ce que vous en choisissiez une. Voir Préparer un travail ; pour les machines, la mise sous tension et la prise d'origine : Machines et mise sous tension.

FAO

FAO (Alt+2) crée les programmes : le bridage, les formes et les opérations du travail à gauche, les champs de l'élément choisi à droite, **Générer le programme** en bas. Voir Créer des programmes. À partir du niveau Ingénieur.

Programme

Programme (Alt+3) contient les programmes et les pièces d'exemple, l'éditeur et un aperçu avec les vérifications du programme. Voir Programmes et éditeur.

Exécution

Exécution (Alt+4) exécute le programme chargé. La colonne de droite montre la position, la broche et l'avance, les corrections, la coupe, les touches du pupitre, le mode manuel, ainsi que les outils et le zéro. Voir Exécuter un programme et le reprendre.

Analyse

Analyse (Alt+5) revient sur les exécutions : leurs courbes, le temps et les efforts de chaque ligne, et les lobes de stabilité. À partir du niveau Ingénieur. Voir Analyse et diagnostic.

Diagnostic

Diagnostic (Alt+6) montre la chaîne d'alimentation de la machine, les entrées du contrôleur et les variateurs, et permet d'injecter des défauts dans le jumeau. À partir du niveau Technicien. Voir Analyse et diagnostic.

Messages et marche à suivre

Studio vous dit ce qui s'est passé, et ce qu'il faut faire, à ces endroits :

- la barre machine : l'étape suivante, ou l'alarme et les étapes du retour à l'état de marche ;
- l'onglet **Messages** du panneau inférieur : les avertissements et les constats ;
- la **Console** : chaque ligne du contrôleur et de Studio ;
- **Outils > Journal d'audit…** (Ctrl+L) : tout, avec qui l'a fait et quand.

Machine > Effectuer l'étape suivante exécute l'étape que montre la barre machine, et seulement celle-là. Quand une alarme a arrêté un programme, les étapes se terminent par la ligne à partir de laquelle reprendre (voir Reprendre à partir d'une ligne).

Niveaux d’utilisateur

Un niveau d’utilisateur montre ce dont une personne a besoin à ce stade et cache le reste. Il y en a cinq :

Niveau	Ajoute
Démo	L’espace de travail **Exécution** avec les pièces d’exemple ; les onglets **Programme** et **Messages**
Opérateur	**Préparation** et **Programme** ; les corrections, le pupitre, le mode manuel, les outils et le zéro ; les sous-menus de **Machine** ; la console, la table d’outils, **Activité IA** ; les paramètres, le journal d’audit, le menu **IA** ; le changement de machine
Technicien	**Diagnostic** ; le positionnement de la pièce ; **Rétablir l’état d’usine…**
Ingénieur	**FAO** et **Analyse** ; les relevés de coupe, les courbes, **Coloration**, l’exportation des courbes, le modèle de pièce à comparer
Chercheur	La carte **Modèle et données** dans **Diagnostic**

Chaque niveau a tout ce qu’ont les niveaux situés au-dessus de lui dans le tableau. Une nouvelle installation démarre au niveau **Ingénieur**.

Pour changer de niveau, utilisez **Outils > Niveau d’interface**, le niveau dans la barre supérieure, ou **Outils > Paramètres… > Général**.

> **Remarque**
>
> Un niveau organise la fenêtre pour l’enseignement et l’apprentissage. Il ne verrouille rien : les touches de la machine fonctionnent à tous les niveaux.

Clair, sombre et mode atelier

Affichage > Apparence choisit **Comme le système**, **Clair** ou **Sombre**, et active ou désactive **Mode atelier (grand format)**. Les deux choix sont indépendants : le mode atelier fonctionne avec l’une ou l’autre apparence.

Les panneaux de Studio lui-même changent aussitôt. Les listes et les zones de texte que dessine le système prennent la nouvelle apparence au prochain démarrage.

Le mode atelier sert à travailler à la machine, à une longueur de bras :

- la barre machine, ses touches et l’arrêt d’urgence sont plus grands, tout comme le texte et le listing ;
- **Exécution** a sa propre disposition : la position, le programme et un grand listing à gauche, les jauges sous la vue 3D, les corrections, les touches du pupitre et les messages à droite, pas de panneau inférieur ;
- la vue 3D a deux touches de vue et pas de gros plan sur l’outil.

Les panneaux et leurs dimensions

Faites glisser le bord entre deux panneaux pour changer leurs dimensions ; double-cliquez sur le bord pour revenir à la dimension d’origine. La vue 3D garde au moins 360 × 240 pixels à l’échelle d’affichage de 100 %. Les bords des panneaux ne se déplacent qu’à la souris. **Affichage > Réinitialiser la disposition** remet d’un coup chaque panneau de chaque espace de travail à sa taille de conception.

Studio retient, pour chaque machine, les dimensions des panneaux, l’onglet du panneau inférieur et son état replié ou non, et l’espace de travail où vous étiez. Il retient la position et la taille de la fenêtre pour le prochain démarrage.

Machines et mise sous tension

Les machines dans Studio, comment passer de l'une à l'autre, et comment une machine est mise sous tension et référencée avant de travailler.

Machines

Chaque machine garde son propre monde : sa position, sa pièce, ses outils et ses décalages d'origine, ses températures et ses heures de fonctionnement. La machine de votre fenêtre est celle qui tourne ; les autres attendent dans l'état où elles ont été enregistrées. En version 1.0, une seule machine tourne à la fois.

Changer de machine

1. Cliquez sur la machine dans la barre supérieure, ou appuyez sur Ctrl+Maj+M (**Fichier > Changer de machine…**).
2. Tapez pour trouver une machine, choisissez-la avec ↑ ↓ et appuyez sur Entrée. Résultat : Studio enregistre la machine que vous quittez et redémarre avec l'autre. Cette machine démarre à froid : mettez-la sous tension et faites la prise d'origine.

Vous ne pouvez pas changer de machine pendant l'exécution d'un programme. La liste montre aussi les **Machines d'entraînement** que les clients IA utilisent en arrière-plan ; **Afficher (lecture seule)** les montre dans **Activité IA**.

La bibliothèque de machines

Fichier > Bibliothèque de machines…, ou un clic sur le logo, montre vos machines avec leur état, leur pièce et leur outil, ainsi que les modèles de machine. Cliquez de nouveau sur le logo pour revenir en arrière.

Pour la machine sélectionnée :

- **Ouvrir** : passer à cette machine.
- **Renommer…** : lui donner un nom de votre choix.
- **Copier comme nouvelle machine…** : une nouvelle machine avec le monde de celle-ci tel qu'il a été enregistré en dernier.
- **Afficher son dossier** : son monde sur le disque.
- **Réinitialisation d'usine… (deux clics en 5 s)** : la machine telle qu'elle est sortie d'usine (voir Repartir de zéro avec une machine).

Nouvelle machine… ajoute une machine du modèle dans son état d'usine. Les nouvelles machines, les copies et les réinitialisations d'usine demandent le niveau Technicien. La ligne **MC-1 réelle** est une place réservée pour plus tard : dans la version 1.0, une machine réelle se connecte depuis la ligne de commande (voir Une machine réelle).

Mise sous tension et prise d'origine

Mise sous tension

Le jumeau possède la chaîne d'alimentation de la MC-1 : l'interrupteur principal QF1 donne 24 V au contrôleur, et **Mettre sous tension** ferme le contacteur KM1, qui donne 48 V aux variateurs et alimente le variateur de broche. L'arrêt d'urgence ouvre KM1.

1. Allez à l'étape **Tension**.
2. Si l'interrupteur principal est sur arrêt, cliquez sur **Enclencher l'interrupteur principal**.
3. Si la touche indique **Enfoncé**, l'arrêt d'urgence est enfoncé : cliquez dessus, ou appuyez sur F1, pour le relâcher.
4. Cliquez sur **Mettre sous tension** (F2). Résultat : le panneau indique 48 V présent et chaque variateur **Prêt**. L'étape suivante est **Origine**.

Prise d'origine

Cliquez sur **Faire la prise d'origine** (F3). Z monte d'abord, puis X et Y ensemble, chacun jusqu'au fin de course à son extrémité de la course ; en temps réel, cela prend environ 20 s.

Résultat : **Référencé : X Y Z**, et les limites logicielles sont actives. Tant que la machine n'est pas référencée après une mise sous tension, le contrôleur est en ALARM:11, et le capteur de longueur d'outil, le palpeur et **Aller au zéro** attendent.

Limites logicielles et course

La MC-1 a une course de 200 mm en X, 150 mm en Y et 80 mm en Z. Après la prise d'origine, la coordonnée machine de chaque axe vaut 0 à son extrémité de référence : X va de −200 à 0, Y de −150 à 0 et Z de −80 à 0, avec 0 en haut de Z.

- Un déplacement manuel qui dépasserait une limite n'est pas exécuté (error:15). Un déplacement manuel maintenu s'arrête juste avant la limite.
- Une ligne de programme qui dépasserait une limite arrête le programme avec ALARM:2. La position est conservée : **Réinitialiser**, **Déverrouiller**, puis vérifiez le programme et l'origine pièce.
- Un axe qui atteint un fin de course déclenche ALARM:1 : **Réinitialiser**, **Déverrouiller**, dégagez l'axe du fin de course par un déplacement manuel, puis **Faire la prise d'origine**.

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Préparer un travail

L'espace de travail **Préparation** prépare la machine et la pièce, une étape à la fois : la mise sous tension, la prise d'origine, le brut et son maintien, les outils et l'origine pièce. Chaque étape apparaît d'abord dans le jumeau, de sorte que vous voyez le bridage avant que quoi que ce soit ne bouge. Les machines, la mise sous tension et la prise d'origine ont un chapitre à part : Machines et mise sous tension.

L'espace de travail Préparation

Préparation (Alt+1) a les étapes à gauche, la vue 3D au milieu et le panneau de l'étape choisie à droite. Le panneau inférieur sous la vue contient la **Table d'outils**.

Les étapes

Étape	Son panneau	Terminée quand
Tension	L'interrupteur principal, **Mettre sous tension**, **Réinitialiser**, **Déverrouiller**	Le contrôleur est en marche et les variateurs sont alimentés
Origine	**Faire la prise d'origine**, **Déverrouiller**, **Réinitialiser**	Les trois axes sont référencés
Bridage	Le brut, son maintien et son emplacement	Un brut est mis en place, ou la table est laissée vide
Outil	L'outil dans la broche, le capteur de longueur d'outil	L'outil de référence est mesuré, une table d'outils est chargée ou l'origine pièce en Z est réglée
Zéro	Le déplacement manuel, l'origine pièce, la carte de palpage	Un décalage d'origine est défini
Programme	**Ouvrir un programme…**, **Pièces d'exemple…**, **Vérifier le programme**	Un programme est chargé
Usinage	**Départ cycle**, **Bloc à bloc**	Le programme est allé jusqu'à sa fin

Une étape terminée porte une coche verte ; l'étape suivante a une petite flèche à son extrémité. Le panneau montre l'étape suivante jusqu'à ce que vous cliquiez sur une autre ; cliquez sur l'étape suivante pour que le panneau la suive de nouveau. La ligne sous les étapes dit ce qu'il faut faire maintenant. Le **?** d'un panneau ouvre sa partie de ce manuel.

La vue 3D pendant la préparation

Pendant que vous modifiez les champs du brut, le brut en cours de préparation est dessiné en transparence sur la table : en vert quand il peut être mis en place ainsi, en rouge quand il ne le peut pas. **Placer** le met en place ; **Restaurer** ou Échap abandonne la modification.

Le brut

Forme, matière et dimensions

Dans l'étape **Bridage**, sur la carte **Pièce et bridage** :

1. Choisissez la **Forme** : **Bloc**, **Barre ronde (verticale)**, **Tube (vertical)**, **Barre hexagonale (verticale)**, **Barre ronde (couchée)** ou **Fichier STL (importer la forme)…**.
2. Choisissez la **Matière**, ou tapez son nom selon une autre norme (C45, EN AW-6061, SUS304) : la liste montre la nuance dont Studio prend les données de coupe, et le bridage garde le nom que vous avez écrit.
3. Saisissez les dimensions que demande la forme : **Longueur L**, **Largeur W**, **Hauteur H**, **Diamètre D**, etc.
4. Choisissez comment il est maintenu et où (voir ci-dessous), puis cliquez sur **Placer**. Résultat : le panneau indique ce qui se trouve sur la table et sa masse.

Les dimensions sont exprimées dans l'unité d'affichage ; en pouces, les champs acceptent aussi les fractions (voir Fractions de pouce).

> Remarque
>
> Un nom issu d'une autre norme est une nuance voisine : les données de coupe viennent de la nuance la plus proche dans la liste. La matière est celle qu'indique le certificat du fournisseur.

Un brut à partir d'un fichier STL

Choisissez **Fichier STL (importer la forme)…** dans la liste des formes et sélectionnez le fichier (en mm). Studio pose la forme sur son point le plus bas et la centre ; sa boîte englobante donne les dimensions. La matière sous un surplomb compte comme pleine. Si le fichier ne peut pas être lu, la console dit pourquoi.

Maintien du brut

Choisissez le **Montage** : **Brides · deux extrémités**, **Brides · deux côtés**, **Brides · quatre côtés**, **Étau**, **Vé (barre couchée)**, **Ruban adhésif double face**, **Non fixé (simplement posé)** ou **Fixation idéale (pour scripts)**.

Le brut et son montage doivent rester sur la table (300 × 220 mm) et à l'écart du capteur de longueur d'outil, à l'arrière gauche. Pendant l'exécution d'un programme, le jumeau compare l'effort de coupe à ce que le montage retient : il avertit quand la marge est faible, et un brut qui glisse bouge dans le jumeau, de sorte que ce qui est usiné ensuite est décalé.

Étau et cales parallèles

Dépassement des mors est la hauteur dont le dessus du brut dépasse les mors : 5 mm sauf si vous la changez. Studio choisit les cales parallèles qui donnent au moins cette hauteur. L'étau s'ouvre de 3,25 à 77 mm, et les mors doivent serrer au moins 2 mm du brut.

Brides

Brides plates sur goujons dans les trous taraudés de la table (M6, au pas de 25 mm) : **deux extrémités** dans le sens de la longueur du brut, **deux côtés** en travers, **quatre côtés** les deux. Chaque bride a besoin d'un trou à 7–35 mm en dehors du bord du brut ; quand il n'y en a pas, le panneau nomme la bride qui ne trouve pas de trou. Utilisez le vé ou l'étau pour une barre couchée.

Ruban adhésif double face

Le ruban maintient le brut par sa surface sur la table. Utilisez-le pour les coupes légères et les pièces plates, pas pour une barre couchée. Une fois qu'il lâche, seul le frottement retient le brut. **Non fixé (simplement posé)** ne tient que par le poids et le frottement ; **Fixation idéale (pour scripts)** ne glisse jamais, pour les scripts et les tests.

Espace au-dessus du brut

Les programmes que la FAO écrit utilisent l'espace au-dessus du brut comme suit :

- Un programme démarre, se termine et change d'outil en haut de la course en Z.
- Entre deux opérations, l'outil se déplace à la hauteur de sécurité : 5 mm au-dessus de la face supérieure du brut, ou 2 mm au-dessus d'une bride plus haute que le brut.
- Là où le bridage laisse moins de place que cela (un outil long, un étau haut, un brut haut), Studio abaisse la hauteur de sécurité à ce qui est disponible, à au moins 0,3 mm sous le haut de la course, et le signale avec le programme. Quand la hauteur de sécurité serait à moins de 0,5 mm au-dessus du brut, Studio n'écrit pas le programme et dit pourquoi.

Le tableau **Dégagement en Z** de la fiche de bridage vérifie le palpeur et chaque outil pour chaque bridage. Voir Dégagement en Z.

Placer le brut

Position X et **Y** placent le milieu du brut dans les coordonnées de table du jumeau (le centre de la table est en X 100, Y 75). **Angle °** le fait tourner dans le sens antihoraire, vu de dessus.

- **Placer** : un nouveau brut à cet endroit. Le brut précédent et ce qui y a été usiné disparaissent.
- **Déplacer ici** : le même brut déplacé, tourné ou maintenu autrement. Ce qui a été usiné reste.
- **Retirer** : le brut ôté de la table.
- **Restaurer** : les champs reviennent à ce qui est sur la table.

La mise en place attend tant que la broche tourne, que la machine se déplace ou qu'un programme s'exécute.

Positionner dans la vue 3D

Cliquez sur **Positionner la pièce** (à partir du niveau Technicien), ou utilisez **Affichage > Vue 3D > Positionner la pièce**. Ensuite :

- faites glisser le brut pour le déplacer, la poignée orange pour le faire tourner, les poignées bleues pour le dimensionner (avec Maj par petits pas, avec Ctrl par grands pas) ;
- ou utilisez les touches : les touches fléchées le déplacent de 1 mm (Maj 0,1 mm, Ctrl 10 mm), Q et E le font tourner de 15° (Maj 1°), Entrée le met en place, Échap le remet comme avant.

Quitter l'espace de travail **Préparation** met fin au positionnement.

Retourner le brut

Retourner retourne le brut dans son montage, de 180° autour de sa longueur : le dessus et le dessous échangent leurs places, de même que l'avant et l'arrière. Ce qui a été usiné tourne avec lui. Les décalages d'origine restent comme ils étaient ; réglez donc de nouveau l'origine pièce (voir Palpage).

Pour une pièce usinée sur deux faces, la FAO prévoit le retournement entre ses phases (voir Travaux à plusieurs phases).

Outils et table d'outils

La **Table d'outils** du panneau inférieur liste les outils de la machine : le numéro T, l'outil et la spécification que lit le programme. L'outil dans la broche porte la mention **Dans la broche**. La table vient avec une pièce d'exemple, un travail FAO ou un client IA ; vous ne pouvez pas la modifier dans la fenêtre.

Avec une table d'outils, le programme change d'outil de lui-même (`M6`) : l'outil se met en place, la machine le mesure sur le capteur de longueur d'outil et continue. Sans table, un `M6` attend que vous mettiez l'outil en place (**Changer d'outil** dans l'étape **Outil**), puis **Départ cycle**.

Changer l'outil dans la broche

1. Allez à l'étape **Outil**.
2. Choisissez l'outil dans la liste **Outil**, par exemple **Fraise 2 tailles Ø6, 2 dents (longueur de sortie 15.5)**.
3. Cliquez sur **Changer d'outil**.

La broche doit être à l'arrêt et la machine au repos. Si le nouvel outil est plus long ou plus court, mesurez-le (**Définir la correction d'outil**) ou réglez de nouveau l'origine pièce en Z. La liste contient les outils standard, métriques d'abord (en pouces d'abord quand Studio affiche des pouces). Comment un outil est décrit, par exemple `flat 6 n2 L15.5` : Outils et table d'outils.

Mesurer les outils avec le capteur de longueur d'outil

Le capteur de longueur d'outil se trouve à l'arrière gauche de la table. La machine doit être référencée et au repos, et la broche arrêtée.

1. L'outil avec lequel vous avez réglé l'origine pièce en Z étant dans la broche, cliquez sur **Mesurer l'outil de référence**.
2. Après chaque changement d'outil, cliquez sur **Définir la correction d'outil**. Résultat : Studio mesure le nouvel outil et définit sa correction de longueur (`G43.1`) à partir de la différence.

Les mêmes touches se trouvent dans **Machine > Capteur de longueur d'outil et palpeur**. La référence est conservée jusqu'à la fermeture de Studio : mesurez-la de nouveau après un redémarrage.

Le modèle de pièce à comparer

Avec le modèle de la pièce, le jumeau montre où la pièce usinée s'en écarte (à partir du niveau Ingénieur).

1. Réglez d'abord l'origine pièce : le modèle est placé à l'origine du programme quand vous le chargez.
2. Dans l'étape **Bridage**, sous **Comparaison au modèle**, cliquez sur **Modèle STL (comparer)…** et choisissez le fichier STL de la pièce (en mm, son origine à l'origine du programme). Résultat : la pièce est colorée selon son écart par rapport au modèle (**Affichage > Coloration > Pièce : écart par rapport au modèle**) : un écart de ±20 µm compte comme bon, les autres couleurs montrent la matière restante et les surcoupes.

Arrêter la comparaison retire le modèle. Les pièces d'exemple chargent le leur.

Enregistrer et ouvrir des bridages

- **Enregistrer le bridage…** écrit le bridage de la carte (la forme, les dimensions, la matière, le montage et l'emplacement) dans un fichier JSON.
- **Ouvrir un bridage…** en relit un dans la carte ; cliquez sur **Placer** pour le mettre en place. Il ouvre aussi un fichier STL comme forme du brut.
- **Exporter la pièce…** enregistre la pièce telle qu'elle est usinée, en STL (en coordonnées programme, brut ou machine) ou en carte de hauteurs 16 bits (PNG). La même commande se trouve dans le menu Fichier : **Fichier > Exporter la pièce…**, depuis n'importe quel espace de travail.

Chaque machine garde son bridage et sa pièce usinée dans son monde. Studio l'enregistre chaque fois que la machine est restée immobile deux secondes après un changement, et à la fermeture de la fenêtre ; après un plantage ou une coupure de courant, la machine revient telle qu'elle était à son dernier arrêt.

Travaux à plusieurs phases

Une pièce usinée sur deux faces, ou maintenue de deux façons, est un seul travail à plusieurs phases. Dans l'espace de travail **FAO**, **Phase…** ajoute une phase au travail. L'arborescence du travail a alors un niveau par phase, avec dessous ses opérations et son programme, et une étape **Retournement** entre deux phases. Un travail à une seule phase n'a pas ce niveau.

- Chaque phase a ses propres opérations et son propre programme. Les programmes portent le nom du travail et le numéro de la phase, par exemple `plate-1.nc` et `plate-2.nc`.
- Une phase part de la pièce telle que les phases précédentes l'ont laissée. Dans le jumeau, Studio exécute la phase 1, retourne la pièce comme le dit l'étape **Retournement**, et écrit le programme de la phase suivante pour cette pièce. Dans la vue 3D, ce qui a été usiné avant et se trouve maintenant dessous est dessiné en tirets.
- L'arborescence appelle une phase recto ou verso d'après le nombre de fois où la pièce a été retournée.
- Sous le panneau d'une opération, **Cette phase** montre le montage de la phase, son zéro, la façon dont la pièce a été retournée et le programme.
- **Résultats de vérification** a une ligne pour chaque phase et une pour le retournement. **Total** additionne le temps à la machine et donne le pire indice de broutement et la plus grande surcoupe. Il compare aussi la pièce terminée, après la dernière phase, au modèle de pièce.

Un travail a au plus 8 phases. Un travail imbriqué ne peut pas encore avoir de phases. Plusieurs phases sont disponibles à partir de l'édition Maker (voir Éditions).

La fiche de bridage a une section pour chaque phase, avec ce qu'il faut faire entre elles. Voir Fiches de bridage.

Retourner la pièce entre deux phases

Sélectionnez **Retournement** dans l'arborescence du travail. Son panneau indique comment la pièce arrive à la phase suivante. Les réglages appartiennent à la phase qui suit l'étape.

Réglage	Choix
Mode de retournement	**Autour de U, 180°** : les extrémités gauche et droite restent en place ; l'avant et l'arrière échangent leurs places, de même que le dessus et le dessous. **Autour de V, 180°** : les bords avant et arrière restent en place ; la gauche et la droite échangent leurs places, de même que le dessus et le dessous. **Sans retournement (autre bridage)** : la pièce reste comme elle est posée, et seul le montage change.
Origine	**Coin supérieur avant gauche** ou **Centre de la face supérieure** : l'endroit où l'origine pièce est retrouvée (voir Reporter l'origine pièce d'une phase à l'autre)
Montage	Comment la pièce est maintenue à partir de cette phase : le même montage, ou un autre

U suit l'axe X de la machine et V son axe Y, en traversant la pièce telle qu'elle est posée dans la phase précédente.

Actions de l'opérateur liste les étapes à la machine : laisser le programme de la phase précédente se terminer (broche arrêtée, liquide de coupe arrêté), desserrer, retourner la pièce, la serrer, régler de nouveau l'origine pièce et charger le programme suivant. **Dans le jumeau** dit ce que le jumeau fait de la pièce.

> **Attention**
>
> Retournez la pièce réelle comme le dit l'étape. Le programme suivant n'est vérifié dans le jumeau que pour la pièce retournée de cette façon. Une pièce retournée autour de l'autre axe, ou pas retournée du tout, est usinée là où rien n'a été vérifié.

Reporter l'origine pièce d'une phase à l'autre

Quand la pièce est retournée ou maintenue dans un autre montage, ses coins ne sont plus là où ils étaient : l'origine pièce (G54) doit être retrouvée. L'arborescence du travail le signale à côté de la phase, et le programme de la phase part de l'origine que vous avez choisie :

- **Coin supérieur avant gauche** : le coin avant gauche du dessus de la pièce, tel que la pièce est maintenant posée. C'est l'origine après un retournement ou un changement de montage, sauf si vous en choisissez une autre.
- **Centre de la face supérieure** : le milieu du dessus de la pièce, par exemple pour une pièce ronde.

Retrouvez l'origine avec le palpeur (voir Palpage). La fiche de bridage donne les étapes de palpage pour chaque phase.

Quand la pièce n'est ni retournée ni maintenue autrement, l'origine de la phase précédente est conservée et n'est pas retrouvée. Une seconde phase qui n'utilise que d'autres outils fonctionne ainsi ; sa section de la fiche de bridage indique que l'origine pièce n'est pas retouchée.

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Palpage et recherche de l'origine pièce

Avec un palpeur dans la broche, Studio trouve un bord, un coin, le centre d'un alésage ou d'un bossage, ou la face supérieure, et en déduit l'origine pièce. La carte de palpage vérifie le cycle par rapport au bridage avant son démarrage, et le jumeau l'exécute comme le ferait la machine.

La carte de palpage se trouve dans **Préparation**, étape **Zéro**, sous les touches de déplacement manuel. Dans la version 1.0, la carte de palpage agit sur le jumeau. Sur une machine réelle, palpez avec la commande de la machine.

Avant de commencer

Il vous faut :

- le brut mis en place sur la table (étape **Bridage**) ;
- le palpeur dans la broche (voir Le palpeur dans la table d'outils) ;
- la broche à l'arrêt.

La machine n'a pas besoin d'être référencée : un cycle qui trouve la machine en alarme ou non référencée la réinitialise et fait d'abord la prise d'origine, et la carte le dit.

> **Attention**
>
> Sur la MC-1, l'étau avec le palpeur de 30 mm ne laisse aucune place au-dessus du brut (voir Espace au-dessus du brut). Maintenez le brut avec du ruban adhésif double face ou sous des brides quand vous palpez.

Le palpeur dans la table d'outils

1. Dans **Préparation**, allez à l'étape **Outil**.
2. Dans **Outil**, choisissez **Palpeur 3D bille Ø2 (longueur de sortie 30, pour le palpage de bord)**.
3. Cliquez sur **Changer d'outil**.

Résultat : la première ligne de la carte de palpage, **Dans la broche**, nomme le palpeur. Quand la broche contient autre chose, la carte dit de mettre d'abord le palpeur en place.

Dans la description d'un outil, le palpeur s'écrit `probe 2 L30` : une bille de 2 mm, dépassant de 30 mm du porte-outil.

Espace au-dessus du brut : le budget en Z

Avec Z en haut de sa course, la pointe du palpeur doit passer au-dessus du point le plus haut du bridage : le dessus du brut, les mors de l'étau ou les brides. Studio le calcule pour l'outil dans la broche et pour chaque outil de la table d'outils chaque fois que le bridage ou les outils changent. Quand un outil ne peut pas passer au-dessus du bridage, l'étape **Bridage** dit de combien il s'en faut et suggère un outil plus court ou un bridage plus bas (des brides au lieu de l'étau).

La carte de palpage vérifie la même chose pour le cycle : elle déplace le palpeur à une hauteur de sécurité au-dessus du brut et du montage, et montre cette hauteur dans les vérifications.

La carte de palpage

De haut en bas :

- **Dans la broche** : le palpeur, et s'il est étalonné (**Étalonné** avec l'heure, ou **Non étalonné**) ;
- **Cycles de palpage** : les six cycles et les champs de celui qui est choisi ;
- les vérifications ;
- **Voir le trajet en 3D** et **Démarrer le palpage…**.

Après un cycle, la carte montre à la place son résultat (voir Définir l'origine pièce à partir de ce qui a été trouvé). Sous la carte de palpage se trouve la carte **Mesurer la pièce**, et en dessous la carte **Rapports** (voir Contrôle).

Les champs numériques acceptent l'unité d'affichage, et en pouces aussi les fractions. Un champ qui ne peut pas être lu devient rouge et dit pourquoi (voir Saisir des nombres).

Les vérifications

La carte vérifie le cycle pendant que vous le réglez, avant que quoi que ce soit ne bouge. Chaque vérification est signalée par ✓ (vert), ! (orange) ou ✕ (rouge). Toute vérification rouge laisse **Démarrer le palpage…** grisé.

Vérification	Vert	Orange ou rouge
Le palpeur	**Palpeur dans la broche, non déclenché**	Pas de palpeur dans la broche ; l'entrée du palpeur est déjà déclenchée
La machine	**Broche à l'arrêt, machine référencée et au repos**	Orange : en alarme ou non référencée, le cycle fait d'abord la prise d'origine. Rouge : la broche tourne, ou la machine est occupée
Les limites logicielles	**Chaque mouvement dans les limites logicielles**	Un mouvement sort de ces limites

Vérification	Vert	Orange ou rouge
La hauteur de sécurité	**Hauteur de sécurité Z …**, et de combien elle est au-dessus du brut et du montage	Pas de place au-dessus du brut pour passer
La profondeur	**Profondeur … sous le dessus**, et, pour un brut dans l'étau, de combien il dépasse les mors	La bille passerait sous les mors, ou trop près du dessous du brut

Quand le jumeau suit une machine réelle, une ligne rouge dit de palper là-bas avec la commande de la machine.

Essayer d'abord un cycle dans le jumeau

Les vérifications viennent d'un essai à blanc : le jumeau calcule le cycle sans rien déplacer. **Voir le trajet en 3D** dessine ses mouvements dans la vue 3D, à la place du trajet du programme, jusqu'à ce que vous quittiez l'étape.

Chaque contact se déroule de la même façon :

1. un déplacement protégé jusqu'au départ (`G38.3`, F600) ;
2. un contact rapide (`G38.2`, F300) ;
3. un dégagement ;
4. un contact lent (F20) : le résultat est l'endroit où celui-ci a touché.

Cycles de palpage

Choisissez un cycle, remplissez ses champs et cliquez sur **Démarrer le palpage…**. Chaque cycle démarre à 3 mm de l'endroit où il attend la face et cherche 5 mm plus loin. **Profondeur** est la distance sous le dessus du brut à laquelle le centre de la bille touche (3 mm sauf si vous la changez). **Départ** : **Depuis le brut** part du brut dans le bridage ; **X Y donnés** part d'un point que vous donnez en coordonnées pièce.

Bord

Trouve une face du brut. **Quelle face** : **Gauche (X−)**, **Droite (X+)**, **Avant (Y−)** ou **Arrière (Y+)**. **Points** : **1 point**, ou **2 points (aussi la rotation)**, qui donne aussi de combien le brut est tourné.

Résultat : **Face X** ou **Face Y**, et avec deux points **Rotation**.

Coin

Trouve un coin à partir de deux faces. **Quel coin** : **Avant gauche**, **Avant droite**, **Arrière gauche** ou **Arrière droite** ; **Extérieur** est le coin du brut, **Intérieur** celui d'une poche (donnez son emplacement avec **X Y donnés**). **Depuis le coin** est la distance au coin à laquelle chaque face est touchée (5 mm). **3 points (aussi la rotation)** touche deux fois la face X et donne aussi la rotation.

Résultat : **Coin X**, **Coin Y**, et avec trois points **Rotation**.

Alésage et bossage

Centre d'alésage palpe un alésage de son milieu vers l'extérieur ; **Centre de bossage** palpe un bossage de l'extérieur, en passant par-dessus son sommet. **Points** : **4 points** ou **8 points**. **Diamètre** est la dimension nominale ; vide, il est repris d'un brut rond. Un brut en bloc demande le diamètre et **X Y donnés**.

Résultat : **Centre X**, **Centre Y** et **Diamètre**.

Face supérieure

Surface Z palpe une fois la face supérieure, au milieu du brut ou aux **X Y donnés**.

Résultat : **Surface Z**.

Définir l'origine pièce à partir de ce qui a été trouvé

La carte de résultat montre chaque valeur en coordonnées machine et pièce, chaque contact, et comment l'étalonnage du palpeur a été appliqué. Quand le brut est tourné, elle dit de combien ; vous pouvez régler l'angle du bridage sur cette valeur dans l'étape **Bridage**.

Pour faire du résultat l'origine pièce :

1. Cliquez sur **Définir le zéro…**
2. Cochez les axes à régler. Les axes que le cycle n'a pas mesurés ne peuvent pas être cochés : un coin ne donne pas de Z. Résultat : la boîte de dialogue montre la ligne qu'elle va envoyer, par exemple `G10 L2 P1 X-129.982 Y95.011`.
3. Cliquez sur **Définir comme zéro G54** (ou comme zéro du décalage d'origine qui était utilisé).

L'origine se place sur ce qui a été trouvé : la face, le coin, le centre ou la surface. **Copier le programme** copie les lignes que le cycle a envoyées. **Mesurer à nouveau** revient aux champs.

> **Remarque**
>
> Les nombres du jumeau montrent la méthode et le déroulement. Ils ne sont pas la précision de mesure de cette machine.

Étalonner le palpeur sur une bague étalon

Un palpeur se déclenche un peu après le contact (pré-course), et pas de la même façon dans toutes les directions. L'étalonnage sur une bague étalon mesure ce décalage, et les cycles suivants sont corrigés en conséquence.

1. Posez la bague étalon sur la table et choisissez **Étalonnage sur bague**.
2. Saisissez le **Diamètre de la bague** tel que l'indique le certificat de la bague.
3. **Position** : **Depuis le brut (l'alésage du tube)** quand la bague est mise en place comme brut, ou **Donnée** quand vous en donnez le centre.
4. Cliquez sur **Démarrer l'étalonnage…** Résultat : **Rayon effectif de la bille**, **Pré-course XY**, **Lobes triples** et **Centre de la bague**.
5. Cliquez sur **Garder pour cette machine**, ou sur **Ne pas garder**.

L'étalonnage appartient à la machine et à ce diamètre de bille, et accompagne l'archive du monde de la machine. Une réinitialisation d'usine l'oublie. Chaque résultat dit quel étalonnage il a utilisé. Tant que vous n'étalonnez pas, la carte affiche **Non étalonné** : un alésage est alors mesuré trop grand de deux pré-courses, et un bossage trop petit de deux pré-courses.

La machine garde un seul étalonnage. Si vous montez un palpeur avec une autre bille (un stylet Ø0,5 pour les petits trous, par exemple), la carte affiche **Non étalonné** et indique la bille et la bague avec lesquelles l'étalonnage de la machine a été fait : étalonnez la nouvelle bille dans la bague étalon. Cet étalonnage remplace le précédent – revenu à la première bille, étalonnez-la à nouveau.

Quand un cycle s'arrête

Quand un cycle ne peut pas se terminer, la carte de résultat dit pourquoi en rouge, la console le consigne, et **Mesurer à nouveau** vous ramène aux champs. Les raisons habituelles :

- un contact n'a rien rencontré dans sa recherche : la face n'est pas là où le cycle l'attendait. Le contrôleur déclenche ALARM:5 ;
- le palpeur a touché quelque chose en chemin vers un contact : quelque chose se trouve là où le brut ne devrait pas être ;
- les contacts ne forment pas un cercle (alésage, bossage, bague).

Après une alarme, suivez les étapes de la barre machine : **Réinitialiser**, puis **Déverrouiller** (voir Alarmes et conduite à tenir). Le palpeur peut se trouver n'importe où quand le cycle s'est arrêté : regardez la vue 3D avant tout déplacement manuel.

> **Attention**
>
> Si le palpeur touche pendant que la machine est en arrêt d'avance, appuyez sur **Réinitialiser** et palpez de nouveau. **Départ cycle** poursuivrait le mouvement au-delà du contact.

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Créer des programmes (FAO)

Studio calcule lui-même les trajectoires d'outil. Vous décrivez la pièce sous forme d'entités : une poche, un contour, des trous, une gravure. Studio choisit les données de coupe avec les modèles physiques du jumeau et écrit le G-code. Il exécute ensuite le programme dans le jumeau avant que vous usiniez.

Ce que fait la FAO

- Les formes sont des données. La forme d'une entité est un rectangle, un cercle, un polygone, un tracé issu d'un fichier SVG ou DXF, un texte ou un maillage (STL, 3MF, OBJ) pour l'usinage 3D.
- Les trajectoires sont calculées. Studio décale la forme, vide la zone et relie les déplacements.
- Le jumeau choisit les données de coupe. La vitesse, l'avance et la profondeur viennent de la matière, puis des modèles du jumeau pour la puissance de la broche, les efforts de coupe, le broutement et la flexion de l'outil. Ce sont des estimations.
- Le jumeau vérifie le programme. Studio peut exécuter le programme dans le jumeau avant que vous usiniez, et modifier ce que l'exécution révèle comme excessif.

Tout s'exécute sur votre ordinateur. Créer un programme ne demande aucune connexion internet.

Le plus rapide est de décrire la pièce à l'assistant IA, par exemple :

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

L'assistant utilise l'outil `program_generate`. Il dit ce qu'il a fait, combien de temps cela prend et ce que le jumeau a trouvé.

Un travail : entités, outils et matière

Un travail est ce que Studio garde au sujet de la pièce :

- son nom, qui est aussi celui du programme, par exemple `bracket.nc` ;
- la matière, par exemple 6061-T6 (les noms d'autres normes, comme EN AW-6061, sont compris) ;
- l'outil dans la broche et la table d'outils ;
- les entités, dans l'ordre où elles sont usinées.

Chaque programme que Studio écrit est conservé comme version, avec le travail qui l'a produit, dans le dossier des programmes. Rien n'est perdu quand vous le refaites : les anciennes versions restent.

L'assistant peut ouvrir un travail et le modifier, y compris celui que vous avez créé dans l'espace de travail CAM, par exemple :

```
Montrez-moi le travail de bracket.nc, puis mettez sa poche à 3 mm de profondeur et supprimez le deuxième perçage.
```

Il utilise l'outil `job_get` pour lire le travail, avec une ligne par opération, et `job_edit` pour changer des valeurs et supprimer, ajouter ou déplacer des opérations. Le travail modifié est refait comme version suivante du programme et vérifié dans le jumeau comme avant. Quand une modification ne peut pas être faite, rien n'est modifié et l'assistant dit pourquoi. Après les essais propres du jumeau (la boucle fermée qui élimine le broutement), le travail que lit l'assistant est celui de la dernière version, avec ce que ces essais ont changé.

Opérations

Opération	Ce qu'elle usine	Outil
Surfaçage	Le dessus du brut, à plat	Fraise 2 tailles
Poche	Une zone jusqu'à une profondeur : rectangle, cercle, polygone ou une forme	Fraise 2 tailles
Contour	Le long d'un profil : à l'extérieur, à l'intérieur ou sur le profil	Fraise 2 tailles
Rainure	Une rainure d'une largeur donnée le long d'une ligne	Fraise 2 tailles
Perçage	Des trous en des points, avec débourrage quand ils sont profonds	Foret
Alésage	Des trous plus grands que l'outil, en hélice	Fraise 2 tailles
Alésage à l'alésoir	Des trous percés mis à la cote	Alésoir
Filetage	Des filetages intérieurs	Fraise à fileter
Gorge	Une gorge annulaire dans la paroi d'un alésage : circlip, joint torique, piège à lumière	Fraise à rainurer en T
Chanfrein	L'arête d'un profil	Fraise à graver en V ou fraise à chanfreiner
Gravure	Du texte ou des lignes	Fraise à graver en V ou petite fraise 2 tailles
Gravure en V	Du texte ou des formes, la profondeur suivant la largeur	Fraise à graver en V
Perlage	Des cercles qui se chevauchent sur une face, le décor des horlogers	Fraise deux tailles de la largeur d'un cercle
Ébauche 3D, finition 3D	Un maillage	Fraise 2 tailles ou fraise hémisphérique

Dans l'espace de travail CAM, le panneau a des formulaires pour le surfaçage, les poches, les contours et les chanfreins (une forme importée, ou un rectangle, un cercle ou un polygone donné dans le panneau), les rainures, le perçage, l'alésage, le fraisage de filets et la gravure. Un travail avec d'autres opérations – celui d'un client d'IA, ou d'un exemple – s'ouvre quand même : ces opérations sont gardées telles que le travail les donne, et générées et vérifiées avec les autres.

Sélectionnées, le panneau liste leurs réglages sous les noms de program_generate (longueurs en mm) ; un client d'IA les modifie, ou le fichier du travail.

Surfaçage, poches et contours

Une poche est vidée par passes. Sa stratégie fixe le trajet dans chaque passe :

- concentrique (par défaut) : des anneaux du centre vers la paroi ;
- zigzag : des allers-retours ;
- adaptatif : un trajet qui garde l'engagement de l'outil constant, pour les poches profondes dans des matières plus dures.

Les îlots d'une poche sont laissés en place. La reprise des restes ne vide que ce qu'un outil plus gros utilisé avant elle a laissé.

Un contour usine le long d'un profil, passe par passe, en avalant. Des attaches peuvent retenir une pièce que le contour découpe.

Rainures

Une rainure est usinée à pleine largeur par passes, puis élargie à sa largeur. Une rainure plus large que l'outil peut être usinée en trochoïdal : de petites boucles le long de sa ligne, pour moins charger l'outil.

Trous : perçage, alésage, alésage à l'alésoir et filetage

- Le perçage utilise le cycle de perçage du contrôleur. Un trou plus profond que trois fois le diamètre du foret est percé avec débourrage.
- L'alésage usine un trou plus grand que la fraise en suivant une hélice. Jusqu'à deux fois le diamètre de l'outil.
- L'alésage à l'alésoir amène un trou percé à la cote avec un alésoir de l'ajustement du trou. Percez d'abord le trou.
- Le fraisage de filets usine un filetage intérieur avec une fraise à fileter, de bas en haut. Les filetages métriques (M8), unifiés (1/4-20 UNC), de tuyauterie (G1/4) et de tuyauterie coniques (1/4 NPT, Rc1/4) sont connus par leur nom.

Chanfreins, gravure et gravure en V

Un chanfrein suit l'arête d'un profil avec une fraise à graver en V, à la largeur que vous donnez. La gravure suit des lignes ou les traits d'un texte à une seule profondeur ; un texte commence à son X, y est centré ou s'y termine (**Alignement** dans le panneau). La gravure en V usine une forme avec une fraise à graver en V : la profondeur suit la largeur de la forme, de sorte que les angles ressortent vifs.

Ébauche et finition 3D

Un maillage (STL, 3MF ou OBJ) est ébauché par passes, puis fini. Les stratégies de finition sont parallèle, Z constant, pentes raides et douces, et crayon.

Outils et table d'outils

Un outil s'écrit avec son type, son diamètre et ce dont il a besoin d'autre, par exemple :

Spécification	Outil
`flat 6 n3 L20`	Fraise 2 tailles, 6 mm, 3 dents, longueur de sortie 20 mm
`ball 3 L12`	Fraise hémisphérique, 3 mm
`drill 3.4 118 L20`	Foret, 3,4 mm, pointe à 118°
`v 6 90 L18`	Fraise à graver en V, 6 mm, 90°
`reamer 6 H7 n6 L32`	Alésoir machine, 6 mm, pour trous H7
`thread 5 60`	Fraise à fileter, 5 mm, profil à 60°
`flat 2 n2 s4 l6 L10`	Fraise 2 tailles, 2 mm, sur une queue de 4 mm qui commence 6 mm au-dessus de la pointe
`thread 2 60 d1.4 n1 s4 l6 L9`	Fraise à fileter, tête de 2 mm, col de 1,4 mm jusqu'à 6 mm, queue de 4 mm au-dessus

Donnez la longueur de sortie (`L`) de chaque outil. Studio vérifie alors que le porte-outil n'atteint pas le brut, et le dégagement en Z de la fiche de bridage peut juger l'outil.

Pour une petite fraise sur queue étagée et pour une fraise pour contre-dépouilles (fraise à fileter, à rainure en T, à queue d'aronde ou sucette), donnez aussi le diamètre de la queue (`s`) et la hauteur au-dessus de la pointe où commence la queue (`l`). Là où la queue frotterait contre les parois de la coupe, Studio ne descend pas plus bas que `l`. Le jumeau vérifie que la queue et le corps du palpeur ne touchent ni la pièce ni le montage.

La table d'outils liste les outils par leurs numéros. Une entité nomme l'outil qu'elle utilise, et Studio ajoute les changements d'outil (`T2 M6`).

Données de coupe issues du jumeau

Studio part des données de coupe de la matière pour l'outil. Il utilise ensuite les modèles du jumeau :

- la puissance de la broche : la coupe en utilise au plus environ 80 % ;
- le broutement : la profondeur et la vitesse restent là où le modèle de stabilité du jumeau prévoit une coupe stable ;
- le montage : l'effort de coupe reste bien en dessous de ce que le montage retient ;
- la flexion de l'outil : les passes de finition restent légères.

Quand la vérification est activée, Studio exécute le programme dans le jumeau. Si l'exécution montre du broutement ou une charge trop élevée, Studio modifie le programme, par exemple avec des passes moins profondes, et réessaie. Chaque version essayée est conservée. La réponse dit ce qui a été changé et pourquoi.

Ce sont des estimations issues de modèles physiques. Observez et écoutez la première coupe sur la machine réelle.

Ajustements et tolérances

Écrivez une dimension avec son ajustement, par exemple `"d": "Ø12 H7"` pour un trou ou `"fit": "g6"` pour un arbre. Studio usine la dimension au milieu de la zone de tolérance de l'ajustement, et finit la paroi en une passe à part.

Les ajustements courants peuvent être nommés : `sliding` (H7/g6), `locating` (H7/h6), `transition` (H7/k6), `press` (H7/p6).

Un petit trou serré gagne à être alésé à l'alésoir plutôt que fraisé. Studio le dit, et vous pouvez le percer, puis l'aléser avec un alésoir de sa classe.

Le jumeau estime le faux-rond de la fraise, et Studio en tient compte sur les dimensions qui ont un ajustement.

Plusieurs phases

Une pièce usinée sur deux faces demande deux phases. Chaque phase a ses propres entités, et indique comment la pièce y arrive : retournée autour de X ou de Y, maintenue autrement, origine pièce retrouvée. Studio écrit un programme pour chaque phase, et vérifie chacun dans le jumeau sur ce que les phases précédentes ont laissé. Comment ajouter des phases et retourner la pièce : voir Travaux à plusieurs phases.

Pièces imbriquées sur une plaque

Plusieurs pièces peuvent être découpées dans une même plaque. Studio les place rapprochées, garde une marge et ajoute des attaches pour que les pièces restent dans la plaque jusqu'à la fin.

Dimensions en pouces et fractions

Avec des unités en pouces, les dimensions peuvent s'écrire sous forme de fractions : `1/4`, `1 1/2`, `3/16"`. Les outils en pouces figurent dans la bibliothèque d'outils, par exemple une fraise 2 tailles de 1/4". Voir Langue et unités.

Vérifier un programme dans le jumeau

Avant d'usiner, exécutez le programme dans le jumeau. Le jumeau usine sa propre pièce et signale les collisions, les surcoupes, le broutement, la charge de la broche et le temps. Voir Simulation.

> **Attention**
>
> Le jumeau vérifie le programme par rapport au bridage que vous lui avez donné. Préparez le jumeau comme l'est la machine réelle : le même brut, le même montage, les mêmes outils et la même origine pièce.

La fiche de bridage

Une fiche de bridage dit à l'opérateur comment préparer le travail : le brut, le montage, les outils, l'origine pièce et le dégagement en Z. Voir Fiches de bridage.

Programmes et éditeur

L'espace de travail **Programme** (Alt+3) regroupe vos programmes et les pièces d'exemple, un éditeur qui vérifie un programme pendant que vous le tapez, et un aperçu de ce que le programme fait à la pièce. Studio conserve une version d'un programme chaque fois que vous l'enregistrez ou le chargez dans la machine, pour que vous puissiez voir ce qui a changé et revenir en arrière.

Ouvrir un programme

L'espace de travail comporte trois parties :

- à gauche, **Programmes** (chaque programme et ses versions) et **Exemples de pièces** ;
- au milieu, l'éditeur ;
- à droite, l'aperçu 3D, **Charger et vérifier** et les constats de la vérification du programme.

Pour ouvrir un programme :

- **Fichier > Ouvrir un programme…** (Ctrl+O) charge un fichier G-code dans la machine, et dans l'éditeur si celui-ci n'a pas de modifications non enregistrées. Studio lit votre fichier et ne l'écrit jamais.
- Déposer un fichier G-code (.nc, .gcode, .ngc, .tap, .txt, .cnc) n'importe où sur la fenêtre fait de même. Pendant qu'un programme tourne, Studio n'ouvre rien et le signale dans la console.
- **Fichier > Programmes récents** liste les huit derniers programmes ouverts depuis un fichier, le plus récent en premier. Un programme déplacé ou supprimé depuis est retiré de la liste quand vous le choisissez ; **Vider la liste** la vide.
- **Fichier > Imprimer le programme…** (Ctrl+P) imprime le programme chargé dans la machine, ses lignes numérotées, en noir sur blanc, avec son nom, la page et la date en haut de chaque page.
- Double-cliquez sur un programme ou une version sous **Programmes** : son texte va dans l'éditeur. Il ne va pas dans la machine tant que vous n'appuyez pas sur **Charger et vérifier**.

L'éditeur et la machine peuvent contenir des programmes différents. Quand la machine a un programme que vous n'avez pas ouvert dans l'éditeur, la ligne d'état l'indique : Ctrl+K charge le programme de l'éditeur dans la machine, Ctrl+Maj+R reprend le programme de la machine dans l'éditeur (vos modifications sont d'abord enregistrées comme une version).

Programmes d'exemple

Studio 1.0 est livré avec quatre pièces d'exemple : une plaque de moteur NEMA 17, une face avant d'instrument, une puce microfluidique à gouttelettes et un panneau allégé isogrid. D'autres arrivent avec les mises à jour. Double-cliquez sur l'une d'elles sous **Exemples de pièces**, ou choisissez-la dans **Fichier > Pièces d'exemple…**.

Charger un exemple prépare toute la machine pour lui : le brut et le montage, la table d'outils et l'outil dans la broche, la prise d'origine, l'origine pièce, le programme, et le modèle de comparaison quand l'exemple en a un.

> **Remarque**
>
> Un exemple remplace le brut, la table d'outils et l'origine pièce G54 que vous aviez. **Programme de démonstration**, dans la même liste, ne charge que le programme et laisse le brut tel qu'il est.

L'éditeur

L'éditeur est un éditeur de G-code :

- les mots sont colorés selon leur type : codes G et M, axes, mots d'arc, avance, vitesse, outil, commentaires, expressions ;
- la marge affiche les numéros de ligne et des repères : les constats de la vérification (erreurs en rouge, avertissements en ambre, remarques en bleu), les points d'arrêt, et la ligne que la machine exécute ;
- après une exécution, une seconde marge affiche le temps de chaque ligne en secondes : en ambre quand l'avance n'a pas été atteinte, en rouge quand il s'y est passé quelque chose (un rapide dans la matière, une coupe avec la broche arrêtée, une collision, une alarme). Cliquez dessus pour voir les chiffres de la ligne lors de la dernière exécution.

Les lignes d'état sous l'éditeur affichent la ligne et la colonne, si le texte est dans la machine, la version, les décomptes de la vérification, et l'état du programme après la ligne où vous vous trouvez : les codes modaux, l'avance, la vitesse, l'outil et la position.

Agrandir donne tout l'espace de travail à l'éditeur ; **Restaurer** ramène les programmes et l'aperçu.

Couleurs et numéros de ligne

Les couleurs suivent l'apparence de la fenêtre (claire ou sombre). Les lignes que le contrôleur a déjà prises en avance, au-delà de la ligne en cours d'exécution, sont grises.

Rechercher et remplacer

- Ctrl+F ouvre la zone de recherche. Entrée trouve l'occurrence suivante, Maj+Entrée la précédente. La recherche ignore la casse et reprend au début une fois la fin atteinte.
- Ctrl+G va à un numéro de ligne.

- Ctrl+H ouvre la zone de remplacement sous la zone de recherche. Entrée remplace l'occurrence sélectionnée et passe à la suivante ; Ctrl+Entrée les remplace toutes. Ctrl+Z annule le remplacement.

Échap ferme les zones.

Points d'arrêt et pauses

Un point d'arrêt arrête le temps du jumeau quand l'exécution arrive à cette ligne ; le mouvement de la machine ne change pas. F5 reprend.

- Ctrl+B, ou un clic dans la marge des repères, place un point d'arrêt sur la ligne.
- **Pause si** (Ctrl+Maj+B) arrête le jumeau quand quelque chose se produit : l'outil commence à couper, un rapide coupe de la matière, une quasi-collision ou une collision, du broutement, une alarme, un changement d'état du contrôleur, l'effort de coupe, la charge de la broche ou l'erreur de poursuite au-delà d'une limite que vous choisissez, ou un changement modal. Quand le jumeau tourne en temps réel, le contrôle du broutement est calculé à côté de la simulation ; une pause sur broutement, et les valeurs de broutement dans les vues, peuvent donc arriver un peu après la coupe : en général quelques centaines de millisecondes, et une ou deux secondes là où beaucoup de nouvelles conditions de coupe se présentent ensemble (une poche autour d'îlots). Tant que le contrôle n'a pas répondu pour un outil, l'indice de broutement affiche **calcul en cours…** ; les courbes et la trace de l'outil attendent la réponse propre à chaque coupe, et montrent donc l'indice de cette coupe. Le rapport de fin d'exécution est le même dans les deux cas.
- Le menu de la ligne (clic droit dans la marge, ou Maj+F10) propose aussi **Exécuter jusqu'à la ligne …** et **Effacer tous les points d'arrêt et conditions**.

Les points d'arrêt prennent effet une fois le programme chargé dans la machine.

La vérification du programme

L'éditeur vérifie le programme pendant que vous tapez (un instant après que vous vous arrêtez), au chargement, et de nouveau après une exécution. Il y a deux parties :

- les règles vérifient chaque ligne par rapport à ce que le contrôleur et la machine acceptent. Elles n'ont besoin de rien d'autre ;
- la vérification de l'aperçu fait tourner le programme sur la pièce telle qu'elle est maintenant. Elle exige que la machine soit sous tension et référencée et qu'aucun programme ne s'exécute. Pendant la saisie, elle prend des programmes de 20 000 lignes au plus ; un programme plus long est vérifié en entier une fois chargé dans la machine, et ses problèmes apparaissent sous forme de cadres rouges dans la vue 3D et dans l'onglet **Messages**.

Programme > Vérifier le programme est une autre vérification : elle fait passer le programme chargé par le mode vérification du contrôleur, ligne par ligne, sans que la machine bouge.

Ce que trouve la vérification

Chaque constat est une erreur, un avertissement ou une remarque, affiché sous sa ligne avec le numéro de sa règle. Exemples :

Règle	Type	Ce qu'elle trouve
S01	Avertissement	Pas de bloc de départ avant le premier mouvement : le programme s'appuie sur l'état du contrôleur
S02	Avertissement	Ni `M2` ni `M30` à la fin
S04	Erreur	Un mouvement de coupe avant tout `F`
S05	Avertissement	La broche démarre sans `S`, ou `S` hors de la plage de la broche
S07	Erreur	Un code que le contrôleur n'accepte pas : expressions, codes `O`, sous-programmes, `G41`/`G42`, `G68`
S10	Avertissement	Un rapide sous le dessus de la pièce et plus profond que ce que le programme a déjà coupé
S11	Erreur	Un arc que le contrôleur refuse
S12	Avertissement	La broche refroidie par eau coupe sans son refroidissement (`M8`)
S14	Avertissement	Un outil d'une autre longueur que celui avec lequel l'origine pièce a été définie, et rien ne le mesure

La vérification de l'aperçu ajoute ce que le programme ferait à la pièce : des rapides dans la matière, une coupe avec la broche arrêtée, des collisions avec le montage, des déplacements au-delà de la course, et une broche qui tourne à sec.

Corriger un constat

Certains constats s'accompagnent d'une suggestion, par exemple **Suggestion (insérer avant) : M8**.

1. Placez le curseur sur la ligne, ou appuyez sur Alt+↓ et Alt+↑ pour passer d'un constat à l'autre.
2. Appuyez sur Ctrl+. (ou choisissez **Appliquer la suggestion** dans le menu de la ligne).

Résultat : la modification se fait en une seule étape ; Ctrl+Z l'annule. Lisez la suggestion avant de l'appliquer : elle corrige le constat, pas votre intention.

Optimiser

Optimiser (Ctrl+Maj+O) cherche un programme plus rapide qui usine la même pièce : il modifie les avances et les vitesses, fusionne les lignes courtes et abaisse les dégagements. Il exécute le programme d'origine et le programme optimisé sur deux copies cachées de la machine, avec la même préparation. Il lui faut un brut mis en place. **Optimiser** fait partie de l'édition Pro et des éditions supérieures.

Si le résultat est meilleur, il remplace le texte de l'éditeur (Ctrl+Z rend l'original) et la comparaison s'ouvre. La console donne les temps, l'effort de coupe maximal et l'indice de broutement avant et après, et ce qui a changé. Appuyez sur **Arrêter l'optimisation** ou sur Ctrl+Maj+O pour l'arrêter.

Versions

Studio enregistre une version d'un programme quand vous :

- appuyez sur **Enregistrer une version** (Ctrl+S) ;
- le chargez dans la machine (**Charger et vérifier**, ou **Fichier > Ouvrir un programme…**) ;
- l'optimisez, ou prenez un programme généré ou celui de la machine à la place de vos modifications (vos modifications sont d'abord enregistrées).

Une version identique à la précédente n'est pas enregistrée de nouveau. Les versions sont conservées dans `Documents\Twinwright Studio\programs\<program>\`, un fichier chacune, et ne sont jamais purgées. **Programmes** les liste sous chaque programme, avec l'heure et la raison de chaque enregistrement.

Comparer deux versions

Comparer (Ctrl+D) affiche, sous l'éditeur, la version enregistrée la plus récente qui diffère du texte de l'éditeur : lignes modifiées en ambre, ajoutées en vert, supprimées en rouge, et lignes où seuls `F` ou `S` ont changé en bleu. Appuyez de nouveau sur Ctrl+D pour une version plus ancienne, sur Alt+↓ et Alt+↑ pour passer d'une différence à l'autre, sur Échap pour fermer.

L'en-tête compte les différences et, quand les deux versions ont été exécutées jusqu'au bout pendant cette session, compare leurs durées d'exécution.

Pour comparer deux versions enregistrées, ouvrez d'abord la plus récente dans l'éditeur, puis appuyez sur Ctrl+D.

Revenir à une version

1. Sous **Programmes**, double-cliquez sur la version. Résultat : son texte va dans l'éditeur. Vos modifications non enregistrées sont d'abord enregistrées comme une version ; Ctrl+Z annule le changement.
2. Appuyez sur **Charger et vérifier** (Ctrl+K) pour la charger dans la machine.

Enregistrer et exporter

- **Enregistrer une version** (Ctrl+S) conserve une version dans le dossier des programmes de Studio.
- Ctrl+Maj+S enregistre le programme de l'éditeur dans un fichier, à l'endroit de votre choix.
- Une version est un simple fichier `.nc` : vous pouvez la copier depuis le dossier des programmes.

Fichier > Exporter les courbes (CSV)… enregistre les courbes de l'exécution (voir Analyse et diagnostic). La fiche de bridage et le rapport de contrôle se trouvent sur la carte **Rapports** de **Préparation** (voir Contrôle).

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Exécuter un programme et le reprendre

L'espace de travail **Exécution** exécute le programme chargé et montre ce que fait la machine pendant la coupe. Quand une exécution s'arrête, Studio vous indique l'étape suivante et peut relancer le programme à partir d'une ligne.

Charger et démarrer

Pour charger un programme dans la machine, procédez de l'une de ces façons :

- **Fichier > Ouvrir un programme…** (Ctrl+O) : choisissez un fichier G-code. Studio le lit ; il n'écrit jamais dans votre fichier.
- **Fichier > Pièces d'exemple…** : choisissez un exemple. Studio met aussi en place pour lui le brut, les outils et l'origine pièce.
- Dans l'espace de travail **Programme**, **Charger et vérifier** (Ctrl+K) charge le programme de l'éditeur.

La barre machine affiche alors le nom du programme, son nombre de lignes et sa durée approximative.

Pour le démarrer, appuyez sur **Départ cycle** (F5) dans la barre machine. **Départ cycle** s'affiche en couleur pleine quand le programme est chargé et que la machine est au repos et référencée. Si la vérification du programme a trouvé des problèmes, la console le signale au démarrage du programme.

Avant d'exécuter un programme sur une vraie machine, vérifiez-le : **Programme > Vérifier le programme** le fait passer par le mode vérification du contrôleur. Le contrôleur lit chaque ligne et la machine ne bouge pas.

Suivre une exécution

Pendant l'exécution d'un programme :

- la vue 3D montre l'outil, la trace de sa pointe et la pièce telle qu'elle est usinée. **Gros plan sur l'outil** montre l'outil à l'endroit de la coupe ;
- l'onglet **Programme** du panneau inférieur affiche le listing : la ligne en cours d'exécution est repérée, les lignes terminées sont plus claires, et une barre indique le temps de chaque ligne. **Suivre la ligne** garde la ligne en vue ;
- **Structure du programme**, à côté du listing, découpe le programme en sections et en passes (d'après les commentaires du programme) avec leurs temps ;
- la colonne de droite affiche les valeurs.

Quand le programme se termine, l'onglet **Messages** indique combien de lignes ont été exécutées et combien de temps l'exécution a duré dans la simulation.

Avancement et temps restant

La barre machine affiche la ligne en cours d'exécution sur l'ensemble des lignes, la part des lignes terminées, ainsi que **écoulé** et **reste** :

- **écoulé** est le temps machine depuis le début, en temps de simulation ;
- **reste** est l'estimation pour la suite, d'après les avances et les rapides du programme.

Après une alarme, la barre affiche la ligne où le programme s'est arrêté.

Les valeurs affichées

La colonne de droite de **Exécution**, de haut en bas :

- **Position** : chaque axe en coordonnées pièce et en coordonnées machine, avec un repère quand l'axe est référencé. Dessous, la ligne modale : les codes G en vigueur, l'outil, l'avance, la vitesse de broche et les codes M. En cas d'alarme, la broche et le refroidissement s'affichent tels que la machine les a, et les valeurs du programme figurent dans l'info-bulle ;
- **Broche et avance** : la vitesse de broche (l'arc est la vitesse, le repère la vitesse demandée), la charge de la broche (verte jusqu'à 80 %, rouge au-delà de 100 %) et l'avance ;
- **Coupe** (à partir du niveau Ingénieur) : les estimations du jumeau pour l'effort de coupe, la puissance, l'indice de broutement et l'avance par dent ;
- **Corrections**, **Pupitre**, **Manuel** et **Outils et origine** (à partir du niveau Opérateur).

Cliquez sur le titre d'une section pour la replier. En mode atelier, **Position** affiche aussi **Reste à parcourir** : la distance qui reste dans la ligne en cours, pour chaque axe.

Arrêt d'avance, arrêt et corrections

- **Arrêt d'avance** (F6 ou Espace) ralentit les axes jusqu'à l'arrêt ; la broche continue de tourner. **Départ cycle** reprend.
- **Arrêt** (F7) commence par un arrêt d'avance, puis réinitialise le contrôleur quand la machine est à l'arrêt. Appuyez de nouveau dessus pour réinitialiser tout de suite.
- **Reset** (F12) arrête le programme immédiatement.

Les corrections modifient l'avance et la vitesse de broche du programme pendant son exécution, de 10 % à 200 % :

Correction	Touches	À l'écran
Avance	Ctrl+↑, Ctrl+↓ : ±10 %	**Corrections** : faites glisser le bouton ; Ctrl pour des pas de 1 %
Broche	Ctrl+→, Ctrl+← : ±10 %	**Corrections** : faites glisser le bouton
Rapide	**Machine > Corrections** : 25 %, 50 %, 100 %	**Corrections** : 100, 50, 25

Ctrl+0, ou un double-clic sur un bouton, remet l'avance et la broche à 100 %.

Bloc à bloc

Bloc à bloc, dans la barre machine, arrête le programme après chaque ligne. Appuyez sur **Départ cycle**, ou sur **Programme > Bloc suivant**, pour exécuter la ligne suivante. Vous pouvez l'activer avant le démarrage ou pendant l'exécution du programme ; cliquez de nouveau dessus pour poursuivre sans arrêts.

Déplacement manuel (jog) et manivelle électronique

Pour déplacer les axes à la main, utilisez **Exécution**, section **Manuel** (repliée au départ), ou les touches (voir Les touches de la machine).

- **Pas** : 0,01 ; 0,1 ; 1 ; 10 ou 50 mm par appui, ou **Maintenir pour déplacer** : l'axe se déplace tant que vous maintenez la touche.
- **Vitesse** : 100, 300, 1500 ou 3000 mm/min.
- Le pavé : X et Y en croix, Z à côté, **Arrêter le déplacement manuel** (Échap) au milieu.
- **Manivelle** : choisissez **X**, **Y** ou **Z** et l'incrément (**×1**, **×10**, **×100** : 0,001 ; 0,01 ou 0,1 mm par cran), puis faites glisser le cadran ou tournez la molette de la souris dessus.

Le déplacement manuel n'est possible que si la machine est sous tension, au repos et sans alarme, et qu'aucun programme ne s'exécute. Un déplacement maintenu s'arrête quand vous relâchez la touche ou quand la fenêtre perd le focus.

Outils > Paramètres… > Panneau et périphériques > Manivelle électronique règle le comportement de la manivelle : **Vitesse** arrête l'axe quand la manivelle s'arrête ; **Trajet** parcourt chaque cran.

Alarmes et conduite à tenir

En cas d'alarme, l'état affiché dans la barre machine devient rouge, et la barre affiche :

- l'alarme et sa signification, par exemple **ALARM:17 Un variateur de moteur signale un défaut** ;
- **Étape suivante :** ce qu'il faut faire maintenant, avec la touche qui le fait ;
- **Toutes les étapes**, quand il y en a plus d'une.

Cliquez sur la touche, ou utilisez **Machine > Effectuer l'étape suivante**, pour effectuer l'étape suivante. Après chaque étape, Studio recalcule les étapes d'après l'état de la machine.

Les autres signes d'une alarme : une ligne sous **Position** quand la position n'est plus fiable, la ligne rouge dans le listing à l'endroit où le programme s'est arrêté, et une étiquette sur le variateur dans la vue 3D.

Les alarmes, leurs causes et ce qui les efface : Codes d'alarme.

Les étapes de rétablissement

Toutes les étapes ouvre les étapes de rétablissement : la cause, puis les étapes dans l'ordre, chacune avec sa touche et la raison pour laquelle elle est nécessaire. L'étape à faire maintenant est repérée. Un clic n'effectue que l'étape suivante.

Par exemple, après une alarme de variateur (ALARM:17) :

1. **Appuyer sur l'arrêt d'urgence, couper l'alimentation des variateurs** (F1) : l'alarme d'un variateur ne s'efface que lorsqu'il perd son alimentation.
2. **Relâcher l'arrêt d'urgence** (F1).
3. **Mettre sous tension** (F2) : les variateurs sont de nouveau alimentés.
4. **Réinitialiser** (F12) : le contrôleur garde ALARM:17 après le rétablissement des variateurs.
5. **Faire la prise d'origine** (F3) : l'alimentation était coupée, la position n'est donc pas fiable.

Quand le défaut a été injecté dans **Diagnostic**, la cause l'indique, avec l'heure.

Quand une alarme a arrêté un programme, les étapes se terminent par la ligne à partir de laquelle redémarrer, par exemple le début de la passe où il s'est arrêté.

Reprendre à partir d'une ligne

Studio peut démarrer un programme à partir de n'importe quelle ligne. Il écrit un bloc de départ qui met la machine dans l'état où le programme l'aurait à cette ligne, puis exécute la suite.

1. Arrêtez le programme. La machine doit être sous tension et aucun programme ne doit s'exécuter.
2. Dans l'onglet **Programme** du panneau inférieur, faites un clic droit sur la ligne et choisissez **Démarrer à la ligne …**, puis **OK**. Dans l'éditeur, faites un clic droit dans la marge de la ligne ou appuyez sur Maj+F10. Résultat : Studio charge « *nom* (depuis la ligne *n*) ». Il ne le démarre pas.
3. Appuyez sur **Départ cycle**.

Le bloc de départ :

1. relève Z tout en haut ;
2. règle les unités, le plan, le mode de distance et le décalage d'origine que le programme avait à cette ligne ;
3. change l'outil, si le programme l'avait changé avant cette ligne ;
4. met en marche le refroidissement et la broche à la vitesse du programme, et attend que la broche atteigne sa vitesse ;
5. se place au-dessus du point de départ à une hauteur de sécurité, descend jusqu'à 2 mm au-dessus de celui-ci, puis descend en avance ;
6. poursuit avec le programme à partir de cette ligne.

La vérification de l'aperçu contrôle ensuite la suite par rapport à la pièce telle qu'elle est.

> **Attention**
>
> Studio ne peut pas suivre certaines lignes avant le départ : paramètres, expressions, sous-programmes, `G10`, `G28`, `G30` et `G92`. Les remarques du chargement les listent. Vérifiez l'état de la machine avant d'appuyer sur **Départ cycle**.

Essayer d'abord le départ sur une copie

Après une alarme qui a arrêté un programme, les étapes se terminent par **Aperçu de la reprise…**. Il ouvre le bloc de départ et ses remarques.

1. Cliquez sur **Essayer sur une copie**. Résultat : le départ s'exécute sur une copie cachée de la machine, à partir de la pièce telle qu'elle est, et Studio indique sa durée, les éventuelles collisions, l'indice de broutement maximal et la quantité de matière enlevée.
2. Si le résultat vous convient, suivez les étapes de rétablissement, puis démarrez à partir de la ligne.

Exécuter à la vitesse de la simulation

Le jumeau tourne en temps réel. **Simulation > Vitesse** le fait tourner de ×0,5 à ×50 le temps réel. L'horloge de la barre machine affiche le temps de la simulation, la vitesse que vous avez choisie et la vitesse que l'ordinateur atteint.

Simulation > Mettre la simulation en pause, ou le sablier de la barre machine, arrête le temps du jumeau : rien ne change tant qu'il est arrêté. **Départ cycle** reprend. Les points d'arrêt et les conditions de pause de l'éditeur arrêtent le temps du jumeau de la même façon (voir Programmes et éditeur).

Mettre la simulation en pause n'est pas un arrêt d'avance : l'état de la machine est figé, pas arrêté.

Analyse et diagnostic

Analyse revient sur une exécution : où le temps est passé, les estimations du jumeau pour l'effort et le broutement, et comment la pièce usinée se compare au modèle. **Diagnostic** montre la chaîne de puissance et les signaux de la machine, et vous permet d'injecter des défauts dans le jumeau pour vous exercer à les trouver.

Examiner une exécution

Analyse (Alt+5, à partir du niveau Ingénieur) se remplit une fois qu'un programme a été exécuté.

- **Exécutions**, à gauche, liste les exécutions de cette session : le programme, la façon dont chacune s'est terminée, son temps machine et son heure de début. Les exécutions qui se sont terminées par une alarme ou une erreur sont en rouge.
- Au milieu se trouvent les courbes, et dessous deux onglets : **Profil par ligne** et **Lobes de stabilité**.
- **Dernière exécution**, à droite, résume la dernière exécution : jusqu'où elle est allée, les lignes qui ont pris le plus de temps, l'effort de coupe et l'indice de broutement les plus élevés avec leurs lignes, la matière enlevée, les lignes où l'avance n'a pas été atteinte, et les événements.

Les courbes, le profil et le résumé sont ceux de la dernière exécution.

Profil par ligne liste les lignes qui ont pris du temps, les plus longues d'abord : le temps et sa part, l'avance demandée et l'avance atteinte, la raison pour laquelle elle n'a pas été atteinte, l'effort de coupe, l'indice de broutement, la matière enlevée et les événements. Double-cliquez sur une ligne, ou appuyez sur Entrée dessus, pour l'atteindre dans l'éditeur **Programme**.

Pourquoi une ligne n'a pas atteint son avance :

- la vitesse maximale des axes ;
- des angles trop rapprochés pour accélérer entre eux ;
- la limite de corde de l'arc ;
- la planification du contrôleur, le logiciel d'envoi ou la lecture du contrôleur qui prennent du retard ;
- une correction d'avance inférieure à 100 %, ou un arrêt d'avance.

Temps : estimé et réel

Avant une exécution, Studio estime le temps machine d'après les avances et les rapides du programme : la barre machine affiche **environ** ce temps, et la **Structure du programme** du panneau inférieur le montre pour chaque section. Pendant et après une exécution, les temps sont le temps machine du jumeau, avec l'accélération et la planification du contrôleur : la barre machine affiche le temps écoulé et le temps restant, **Structure du programme** le temps de toute l'exécution, et **Profil par ligne** le temps de chaque ligne.

Tous les temps sont du temps machine dans la simulation, pas le temps de votre horloge.

Effort, broutement et épaisseur de copeau

Les courbes montrent les valeurs de l'exécution sur les 10, 30 ou 120 dernières secondes : l'avance, la vitesse réelle, la vitesse et la charge de la broche, l'effort de coupe, le couple et la puissance, l'épaisseur de copeau, l'indice de broutement et les erreurs de poursuite. Cliquez sur le nom d'une courbe pour l'afficher ou la masquer.

L'indice de broutement est la profondeur de passe divisée par la profondeur stable à la vitesse de broche utilisée :

Indice	Signification
Moins de 0,8	Stable
0,8–1	Près de la limite
Plus de 1	Broutement

Lobes de stabilité trace la profondeur de passe stable en fonction de la vitesse de broche, pour la dernière coupe (ou, avant toute coupe, pour un rainurage pleine fraise de 1 mm de profondeur suivant X). Cliquez sur **Calculer les lobes de stabilité**. Un point rouge marque la coupe telle qu'elle a été faite, une ligne verte en tirets la vitesse la plus stable.

Affichage > Coloration colore la trace de l'outil selon l'avance, l'effort de coupe, l'épaisseur de copeau, l'indice de broutement ou la ligne du programme, et la pièce selon la matière et la profondeur, l'écart par rapport au modèle ou la ligne du programme.

Ce sont les estimations du jumeau, tirées d'un modèle physique (voir Modèles du jumeau).

Comparer avec le modèle de la pièce

Pour voir où la pièce usinée diffère du modèle :

1. Dans **Préparation**, étape **Bridage**, cliquez sur **Modèle STL (comparer)…** et choisissez le fichier STL de la pièce (en mm, son origine au zéro du programme).
2. Exécutez le programme.
3. Choisissez **Affichage > Coloration > Pièce : écart par rapport au modèle**.

Résultat : la pièce est colorée selon la distance de chaque point au modèle : à ±20 µm près, cela compte comme bon ; les autres couleurs montrent la matière restante et les surcoupes. **Arrêter la comparaison** retire le modèle. Les pièces d'exemple chargent leur propre modèle.

Comparer deux exécutions

Analyse montre la dernière exécution. Pour comparer deux exécutions d'un programme :

- dans l'éditeur **Programme**, **Comparer** (Ctrl+D) donne les durées d'exécution des deux versions et le temps des lignes modifiées, quand les deux ont été exécutées jusqu'au bout pendant cette session (voir Comparer deux versions) ;
- l'assistant IA peut dessiner deux exécutions côte à côte (voir L'assistant IA).

Exporter

Exporter les courbes (CSV)… (dans **Dernière exécution**, ou **Fichier > Exporter les courbes (CSV)…**) enregistre environ les 160 dernières secondes des courbes dans un fichier CSV : une colonne de temps en secondes, puis une colonne par courbe, avec son unité dans l'en-tête et les valeurs dans les unités d'affichage.

Exporter la pièce… dans **Préparation**, ou **Fichier > Exporter la pièce…**, enregistre la pièce usinée sous forme de fichier STL (en coordonnées du programme, du brut ou de la machine) ou de carte de hauteurs.

L'espace de travail Diagnostic

Diagnostic (Alt+6, à partir du niveau Technicien) montre la chaîne de puissance de la machine sous forme de schéma : la puissance circule de gauche à droite. Les traits pleins transportent la puissance, en vert quand ils sont sous tension ; les traits en tirets sont des signaux. Chaque élément est vert quand il est en bon état, gris quand il est éteint, ambre pour un avertissement et rouge pour une alarme ou un défaut.

Cliquez sur un élément pour voir ses détails dans la colonne de droite. Sous le schéma se trouvent les entrées du contrôleur, l'injection de défauts et les alarmes des variateurs.

La chaîne de puissance

La chaîne de la MC-1, du secteur à la broche :

- l'**Interrupteur principal QF1** et l'**Alimentation 24 V** du contrôleur ;
- le **Contrôleur** : son état ou son alarme ;
- l'**Arrêt d'urgence SB0** et **KM1**, le contacteur qu'il ouvre ;
- l'**Alimentation 48 V** des variateurs ;
- les variateurs **X**, **Y** et **Z** et leurs axes ;
- le **Variateur de fréquence**, la **Broche** et la **Pompe d'arrosage**.

Les entrées du contrôleur

Un voyant par entrée : les fins de course (**Fin de course X−**, **Fin de course X+**, **Fin de course Y−**, **Fin de course Y+**, **Fin de course Z**), le **Capteur de longueur d'outil**, l'**Arrêt d'urgence** et l'**Alarme variateur**. Un voyant est gris quand l'entrée est inactive, et ambre ou rouge quand elle est active.

Le voyant **Capteur de longueur d'outil** est l'entrée de palpage du contrôleur : il s'allume aussi quand le palpeur touche.

Les variateurs

Chaque variateur affiche **Prêt** et son erreur de poursuite, ou **Désactivé**, **Alarme**, **Démarrage** ou **Hors tension**. Les détails donnent l'alimentation des variateurs et l'erreur de poursuite de chaque axe, avec sa valeur maximale.

Le variateur de broche (variateur de fréquence)

Le **Variateur de fréquence** affiche sa fréquence et son courant de sortie, ou son défaut : **OC** (surintensité), **OL** (surcharge) ou **LU** (sous-tension). Les détails donnent la vitesse de broche et la vitesse demandée, la charge et la pompe. Une broche qui a tourné sans son refroidissement est signalée parmi les avertissements, avec la durée de marche à sec.

Injecter des défauts dans le jumeau

L'injection de défauts vous permet de vous exercer à trouver et à éliminer des défauts sur le jumeau ; la machine de votre fenêtre est modifiée, aucune machine réelle ne l'est.

- **Fils** : réglez n'importe quel fil de la machine (les fins de course, le capteur de longueur d'outil, la chaîne d'alarme des variateurs, la ligne d'arrêt d'urgence, le circuit KM1, le bus RS485, le relais de défaut du variateur de fréquence) sur **Coupure**, **Court-circuit**, **Intermittent** ou **Parasites** (une seule micro-coupure de 50 µs). **Normal** remet le fil en état.
- **Alarme variateur** : **X**, **Y** ou **Z** fait signaler un défaut à ce variateur. L'alarme d'un variateur ne s'efface que lorsque le variateur perd son alimentation : suivez les étapes de rétablissement dans la barre machine.

Chaque défaut injecté est listé parmi les avertissements, écrit dans la console et enregistré dans le journal d'audit. Après une alarme de variateur, la cause indiquée dans la barre machine nomme le défaut que vous avez injecté et l'heure.

Alarmes : chacune et son rétablissement

Toutes les alarmes, leur signification et leurs causes : Codes d'alarme. La barre machine donne les étapes de rétablissement pour l'alarme en cours (voir Les étapes de rétablissement) :

Alarme	Étapes de rétablissement
Défaut d'un variateur (ALARM:17 avec un variateur en alarme)	Appuyez sur l'arrêt d'urgence, relâchez-le, **Mettre sous tension**, **Réinitialiser**, **Faire la prise d'origine**
Arrêt d'urgence enfoncé (ALARM:10)	Relâchez l'arrêt d'urgence, **Mettre sous tension**, **Réinitialiser**, **Faire la prise d'origine**
ALARM:17 avec les variateurs rétablis	**Réinitialiser**, **Faire la prise d'origine**
Prise d'origine requise (ALARM:11)	**Faire la prise d'origine**
Fin de course matériel (ALARM:1)	**Réinitialiser**, **Déverrouiller**, dégagez le fin de course par déplacement manuel (jog), **Faire la prise d'origine**
ALARM:3, 8, 9	**Réinitialiser**, **Faire la prise d'origine**
Toute autre alarme	**Réinitialiser**, **Déverrouiller**

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Exécuter un programme dans le jumeau

Ébauche (itération 14) : le texte est en cours de rédaction ; les sections peuvent encore s'étoffer.

Ce qu'est le jumeau

La machine de la fenêtre est un jumeau numérique de la MC-1 : le noyau propre à Twinwright y tourne, avec le même code que sur la machine, et un modèle physique représente les moteurs, les variateurs, la broche, les interrupteurs et la coupe. Un programme s'y exécute comme il le ferait sur la machine, et le jumeau usine sa propre pièce. Chaque chiffre qu'il donne sur le monde physique est une estimation.

Avant d'exécuter

Le jumeau vérifie un programme par rapport à la préparation que vous lui donnez ; préparez-le donc comme la vraie machine :

- la machine (**Fichier > Changer de machine…**) ;
- le brut et son maintien (l'espace de travail **Préparation**) ;
- les outils (la **Table d'outils** du panneau inférieur) ;
- l'origine pièce (**Machine > Origine pièce**).

Fichier > Pièces d'exemple… prépare tout cela et charge un programme en une seule fois.

> **Attention**
>
> Une vérification ne vaut que ce que vaut la préparation : un autre brut, un autre montage, un autre outil ou une autre origine pièce sur la vraie machine n'est pas ce que le jumeau a vérifié.

Exécuter un programme

1. Ouvrez le programme : **Fichier > Ouvrir un programme…**
2. Vérifiez-le d'abord : **Programme > Vérifier le programme** le fait passer par le mode vérification du contrôleur ; le contrôleur lit donc chaque ligne et la machine ne bouge pas.
3. Exécutez-le : **Départ cycle** (F5). **Arrêt d'avance** (F6) le met en pause, **Départ cycle** le reprend, **Arrêt** (F7) l'arrête. **Bloc à bloc** l'exécute ligne par ligne, **Bloc suivant** exécute la ligne suivante.

Le jumeau tourne en temps réel, sauf si vous choisissez autrement : **Simulation > Vitesse** le fait tourner de la moitié à 200 fois le temps réel, ou à la vitesse **Maximale** – aussi vite que le permet cet ordinateur – et **Simulation > Mettre la simulation en pause** arrête le temps du jumeau (l'état de la machine ne change pas pendant ce temps). Un clic sur l'horloge, à droite de la barre machine, ouvre la même liste.

À partir de ×100, à la vitesse **Maximale** et pendant un calcul jusqu'à la fin, le jumeau travaille en mode **Allégé** (l'étiquette à côté de l'horloge) : la vérification du broutement et les efforts peuvent arriver un peu plus tard dans les courbes, l'image utilise des cellules plus grossières et une trace plus fine. Ce que fait le programme – la matière enlevée, les collisions, les limites logicielles, les alarmes – est jugé comme à ×1, et le résultat et le rapport sont les mêmes. **Paramètres > Général > Pleine précision même à vitesse élevée (plus lent)** désactive le mode Allégé.

Quand l'ordinateur ne parvient pas à suivre, l'horloge indique en ambre la vitesse à laquelle le jumeau tourne réellement ; son info-bulle nomme la partie du jumeau la plus occupée (par exemple l'enlèvement de matière sur des mailles fines) et ce qu'il faut faire.

Simulation > Vitesse > Calculer jusqu'à la fin (sans lecture) exécute le programme chargé jusqu'à sa fin sans le montrer image par image : la barre machine montre où il en est et le temps restant, la vue 3D reste au début et change une fois le calcul terminé, et la dernière exécution de l'espace de travail **Analyse** est renseignée comme d'habitude. **Arrêter**, à côté du programme, l'arrête là où il en est. Quand le jumeau est connecté à une machine réelle (mode ombre), seul ×1 est possible : le jumeau suit le temps de la machine.

Ce que vérifie le jumeau

Ce qui découle du programme et de la géométrie de la préparation :

- les alarmes et erreurs propres au contrôleur, telles que le contrôleur de la machine les donne : limites logicielles, fins de course, un mouvement de palpage qui ne touche rien ;
- les collisions du porte-outil, de l'écrou de pince et de la broche avec le brut, le montage et la table, et les quasi-collisions ;
- la coupe en déplacement rapide (`G0`), et la coupe pendant que la broche est arrêtée ;
- la coupe à travers le fond du brut jusque dans le montage ou la table ;
- un outil qui se bloquerait ou casserait : une avance par dent trop élevée pour l'outil, une contrainte de flexion à la base des goujures supérieure à ce que l'outil supporte ;
- le dégagement en Z : si chaque outil, et le palpeur, passe au-dessus du bridage avec Z en haut (voir Matériel de la MC-1) ;
- avec le modèle de la pièce dans la préparation : la matière restante sur la pièce et les coupes trop profondes (**Affichage > Coloration > Pièce : écart par rapport au modèle**).

Les constats apparaissent dans l'onglet **Messages** du panneau inférieur au fur et à mesure.

Ce que le jumeau estime

Ce que calcule le modèle physique :

- les efforts de coupe, le couple et la puissance, et la charge de la broche : la broche ralentit en charge comme une vraie, et une charge trop forte met son variateur en défaut ;
- la façon dont l'outil et la machine fléchissent sous la coupe, dans la forme usinée ;
- le broutement : l'indice le long de la trajectoire (**Affichage > Coloration > Trace : indice de broutement**), et les vitesses de broche où la coupe est stable (l'espace de travail **Analyse** : **Calculer les lobes de stabilité**) ;
- le faux-rond de l'outil : les rainures et les trous en sortent plus larges ;
- le temps machine, d'après la planification propre au contrôleur et le mouvement des variateurs ;
- les crêtes qu'une fraise à bout sphérique laisse entre ses passes (leur forme, pas la rugosité de la surface).

La trace peut aussi être colorée selon l'avance, l'effort de coupe ou l'épaisseur de copeau (**Affichage > Coloration**), et l'onglet **Courbes** du panneau inférieur trace l'exécution dans le temps.

> **Remarque**
>
> Les estimations utilisent des paramètres typiques – données de coupe publiées, outils en carbure affûtés, valeurs de conception de la machine – tant que votre machine n'a pas été mesurée. Servez-vous-en pour comparer des programmes et trouver des problèmes avant de couper, pas comme une mesure de la pièce.

Ce que le jumeau ne modélise pas

- la texture et la rugosité de la surface ;
- l'arête rapportée, les copeaux et leur évacuation ;
- les parois minces qui fléchissent sous la coupe ;
- la chaleur de la pièce elle-même ;
- l'usure des outils : la coupe est toujours celle d'un outil affûté. Le jumeau tient une estimation de l'usure de chaque outil – combien de temps ses arêtes ont été dans la matière, quelle longueur elles ont coupée, quelle part d'une durée de vie estimée est consommée (loi de Taylor ; les valeurs de la plupart des matières sont des estimations) – et la signale (tool_wear de machine_status par MCP), mais elle ne change pas la coupe ; un outil neuf de même spécification repart de zéro (setup_tool avec new). Les outils qui usinent l'aluminium sont le plus souvent changés avant d'être usés, parce que l'aluminium colle à l'arête et que la pièce présente des bavures ou une finition moins bonne ;
- ce qui se passe après la rupture d'un outil ;
- les détails plus fins que la grille de la vérification du programme : avant toute exécution, la vérification du programme examine la pièce par cellules d'environ 0,1 mm, plus fines quand le programme a un outil de moins de 2 mm de diamètre : un vingtième de son diamètre, jusqu'à 0,05 mm (plus grossières sur les bruts de plus de 100 × 100 mm : un million de cellules au plus, quatre millions avec un tel outil), si bien qu'un déplacement rapide à travers une paroi plus mince, ou un outil qui effleure une paroi de moins, peut passer la vérification sans être remarqué. Exécutez le programme dans le jumeau pour le voir : la coupe du jumeau lui-même travaille par cellules de 0,02 mm.

Plastiques et micro-outils

En usinant du PMMA (acrylique) avec des outils de 0,1–0,5 mm (des canaux microfluidiques, par exemple), le jumeau laisse aussi de côté :

- la température de la coupe : le PMMA se ramollit à l'approche de sa transition vitreuse (environ 105–115 °C), et dès environ 70 °C ses copeaux se soudent et s'étalent sur l'outil et dans le canal. Restez dans la plage d'avance par dent et de vitesse que donne le fabricant de l'outil ;
- le jet d'air ou le liquide de coupe, et ce qu'ils font à la chaleur et aux copeaux ;
- les bavures le long des bords des canaux et des poches ;
- les parois minces : dans le PMMA, les parois de moins d'environ 0,15 mm d'épaisseur fléchissent ou cassent sous la coupe ; le jumeau usine la forme que laissent la trajectoire de l'outil et sa flexion, pas la déformation de la paroi elle-même ;
- l'amortissement de la coupe elle-même : un outil aussi petit fléchit à des dizaines de kilohertz et coupe lentement, et sa face de dépouille, en frottant sur la surface usinée, amortit sa vibration. Le jumeau l'ignore, si bien que son indice de broutement pour ces outils est du côté de la sécurité.

Les modèles et ce qu'ils supposent : Modèles du jumeau.

Les résultats et le rapport

- La pièce dans la vue 3D est ce que le jumeau a usiné ; sa coloration montre la profondeur de la coupe ou, avec le modèle de la pièce, l'écart par rapport à celui-ci.
- L'onglet **Messages** du panneau inférieur liste les constats des vérifications ; son onglet **Courbes**, les efforts, la charge et les vitesses dans le temps. **Fichier > Exporter les courbes (CSV)…** les enregistre.
- Un client IA lit les mêmes résultats par les outils MCP de Studio : voir la référence Outils MCP.

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Contrôle, rapports et fiches de bridage

Après l'usinage d'une pièce, le palpeur la mesure par rapport à la conception. Studio consigne ce qu'il a mesuré, avec l'estimation du jumeau à côté, dans un rapport de contrôle. Avant l'usinage d'une pièce, une fiche de bridage dit à l'opérateur comment préparer le travail. Les deux sont dans l'édition Pro.

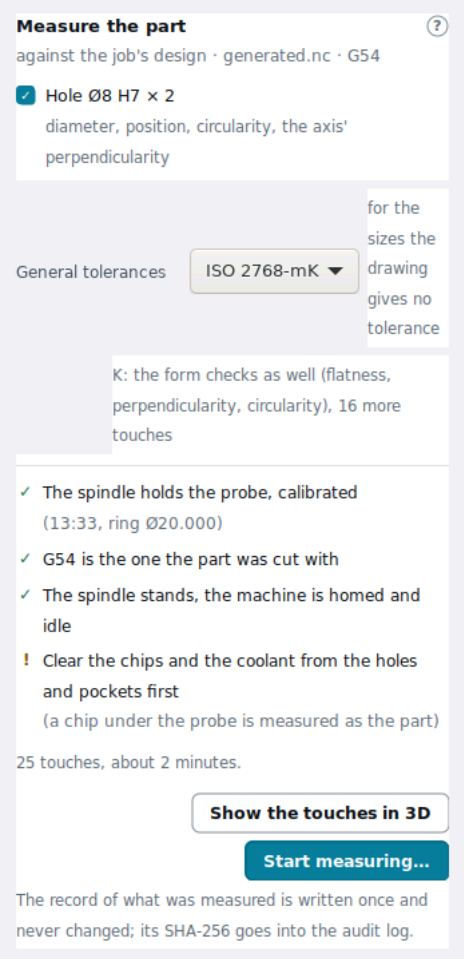
Mesurer une pièce avec le palpeur

Mesurez une pièce là où elle a été usinée, avant de la retirer de la machine. La carte **Mesurer la pièce** se trouve dans **Préparation**, étape **Zéro**, sous la carte de palpage. Elle compare la pièce à la conception de son travail FAO : la dimension, l'ajustement et la tolérance de chaque entité. Dans la version 1.0, la carte mesure sur le jumeau, comme le fait la carte de palpage. Quand le jumeau suit une machine réelle, mesurez là-bas avec la commande de la machine.

Il vous faut :

- La pièce, usinée et toujours tenue comme elle l'était pendant l'usinage. Ne déplacez pas l'origine pièce : les positions sont mesurées à partir d'elle.
- Le palpeur dans la broche, étalonné sur sa bague étalon. Voir Palpage.
- Le travail FAO du programme : un programme créé dans **FAO**, ou par l'assistant IA.

> **Remarque**
>
> La mesure est dans l'édition Pro. Dans les autres éditions, la carte s'affiche avec une étiquette **Pro**, et **Démarrer la mesure…** dit comment l'essayer. Les enregistrements et les rapports créés auparavant s'ouvrent dans toutes les éditions.



Pour mesurer :

1. Dans **Préparation**, cliquez sur l'étape **Zéro**. Résultat : la carte dit à quoi elle compare la pièce : le programme et le décalage d'origine avec lesquels la pièce a été usinée, par exemple G54.
2. Choisissez les entités à mesurer. Toutes sont choisies au départ. Résultat : chaque entité dit ce que le palpeur mesure sur elle. Une entité que le palpeur ne peut pas mesurer est grisée et dit pourquoi.
3. Sous **Tolérances générales**, choisissez la classe pour les dimensions qui n'ont pas de tolérance propre. Ajoutez H, K ou L pour les contrôles géométriques (voir Tolérances). Résultat : la carte dit combien de contacts elle effectue et combien de temps ils prennent à peu près.
4. Lisez les vérifications : le palpeur dans la broche et étalonné ; l'origine pièce, celle avec laquelle la pièce a été usinée ; la broche à l'arrêt ; la machine référencée et au repos.
5. Retirez les copeaux et le liquide de coupe des trous et des poches. Un copeau sous le palpeur est mesuré comme la pièce.
6. Pour voir où le palpeur touchera, cliquez sur **Afficher les contacts en 3D**. Résultat : la vue 3D montre les contacts latéraux et les contacts vers le bas.

7. Cliquez sur **Démarrer la mesure…**. Résultat : le palpeur touche chaque entité, et la carte dit ce qu'il mesure en ce moment.

> ⚠ **Avertissement**
>
> Le palpeur se déplace à des vitesses d'avance près de la pièce. Gardez les mains et les outils éloignés de la machine pendant la mesure.

Ce qui est mesuré :

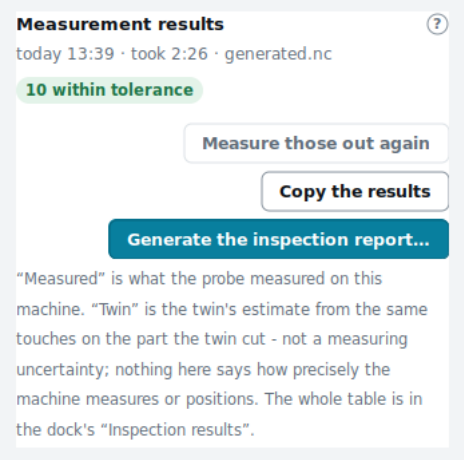
Entité	Contrôles
Poche, contour	Dimension et position de ses parois, profondeur de son fond
Trou, bossage	Diamètre et position
Rainure	Largeur et profondeur
Surfaçage	Profondeur
Chanfrein	Largeur, profondeur et angle
Rainure en V (gravure ou gravure en V avec une fraise à graver en V)	Largeur, profondeur, angle et décalage par rapport à sa ligne
Gravure avec une fraise 2 tailles, filetage, surface 3D	Non mesuré

Les contrôles géométriques ajoutent la planéité, la perpendicularité, la circularité et la rectitude là où elles s'appliquent (voir Forme).

Quand le palpeur a terminé, la carte **Résultats de mesure** prend la place de la carte de mesure. Elle affiche :

- quand la pièce a été mesurée, combien de temps cela a pris, et le programme ;
- combien de contrôles sont dans la tolérance, combien hors tolérance, et combien ne sont pas évalués : **non évalué dans le jumeau** (ses cellules y sont plus grandes qu'un quart de la tolérance), **sans tolérance** ou **non mesuré** ;
- chaque contrôle hors tolérance, avec sa valeur mesurée, sa limite et de combien il est au-dessus ou en dessous ;
- pour la première entité hors tolérance, la cause la plus probable et ce qu'il faut faire ensuite.

Dans la version 1.0, le palpeur mesure la pièce que le jumeau a usinée ; la cause vient donc du jumeau seul. Si la pièce du jumeau est elle-même hors tolérance aux mêmes contacts, c'est le programme qui l'usine ainsi : modifiez-le dans **FAO**. Sinon, la valeur mesurée et l'estimation du jumeau ne diffèrent que par ce que le jumeau modélise du palpeur (le rayon effectif de la bille et, pour un palpeur courant, sa pré-course, qu'un étalonnage sur la bague étalon élimine) et par la taille des cellules du jumeau à l'endroit où le palpeur touche. Étalonnez le palpeur sur la bague étalon et mesurez à nouveau. Les causes sont des suggestions, pas un diagnostic.



Les boutons de la carte :

- **Remesurer les points hors tolérance** : mesure de nouveau les contrôles hors tolérance, par exemple après avoir étalonné de nouveau le palpeur.
- **Copier les résultats** : copie le tableau sous forme de texte, une colonne par tabulation, pour un tableur.
- **Générer le rapport de contrôle…** : écrit le rapport (voir Le rapport de contrôle).

Le tableau complet se trouve dans l'onglet **Résultats du contrôle** du panneau inférieur. Chaque ligne est un contrôle, avec sa valeur **Nominal**, sa **Tolérance**, ce qui a été **Mesuré**, l'**Écart**, l'estimation du **Jumeau** et le verdict : ✓, ou *X* avec l'ampleur du dépassement. Les lignes hors tolérance sont en rouge, et la vue 3D marque les résultats sur la pièce.

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	✓

Chaque contrôle est conservé sous forme d'enregistrement. L'enregistrement est écrit une seule fois et jamais modifié, et son SHA-256 est inscrit dans le journal d'audit (voir Vérifier une copie par rapport au journal d'audit).

L'assistant IA mesure de la même façon, avec l'outil `inspect_part`. Demandez, par exemple : « Contrôlez la pièce de bracket.nc avec le palpeur ».

1. L'assistant dit ce qu'il va mesurer, et combien de contacts et de secondes les contrôles géométriques ajoutent.
2. Confirmez avant que le palpeur ne bouge. Résultat : la réponse liste chaque contrôle avec sa valeur mesurée, l'estimation du jumeau et le verdict.

Tolérances

Chaque contrôle est jugé par rapport à une tolérance. La tolérance propre d'une dimension, ou son ajustement, vient du travail FAO, tel qu'il a été saisi dans le champ de la dimension (par exemple 8 H7). Les autres dimensions prennent la tolérance générale.

Tolérances générales (ISO 2768-1)

ISO 2768-1 comporte quatre classes : f (fine), m (moyenne), c (grossière) et v (très grossière). Une dimension sans tolérance propre prend la tolérance de la classe pour sa plage de dimensions. Choisissez la classe sur la carte, sous **Tolérances générales**. Ou demandez-la à l'assistant, par exemple : « Contrôlez-la selon ISO 2768-m ».

Forme : planéité, perpendicularité, circularité, rectitude (ISO 2768-2)

Ajoutez H, K ou L après la classe, par exemple « mK ». Le palpeur contrôle alors aussi la forme :

Contrôle	Comment il est mesuré
Planéité	Un fond palpé sur une grille de 3 × 3
Perpendicularité	Une paroi, ou l'axe d'un trou, palpée de nouveau à une seconde profondeur
Circularité	Un trou ou un bossage palpé en 8 points
Rectitude	Une longue paroi palpée en 3 à 8 points

Avant que le palpeur ne bouge, la carte, ou l'assistant, dit combien de contacts de plus les contrôles géométriques demandent.

> **Remarque**
>
> Un palpeur qui n'est pas étalonné dans chaque direction a ses propres lobes. Le rapport indique l'ampleur que le modèle de palpeur du jumeau leur attribue, à côté de chaque circularité.

Ajustements (ISO 286)

Une dimension écrite avec son ajustement, par exemple Ø8 H7, est jugée par rapport à la zone de l'ajustement. Voir Ajustements et tolérances.

Chanfreins et rainures en V

Les largeurs, profondeurs et angles de chanfrein, et les largeurs, profondeurs et angles de rainure en V, prennent la classe f de l'ISO 2768-1, sauf si vous donnez leur propre tolérance.

Ce que le jumeau estime

Le jumeau effectue les mêmes contacts sur sa pièce simulée. Son estimation est ce que le programme a usiné dans le jumeau. Quand la valeur mesurée et l'estimation diffèrent, examinez la machine et le bridage : la longueur de l'outil, l'origine pièce, le montage, un outil usé.

L'estimation du jumeau est un calcul issu de ses modèles physiques. Ce n'est pas l'incertitude de mesure de la machine.

Le rapport de contrôle

Chaque contrôle est conservé sous forme d'enregistrement. À partir d'un enregistrement, Studio écrit le rapport :

- un fichier CSV (UTF-8), pour un tableur ;
- un fichier PDF, pour l'impression et l'archivage.

Le rapport indique le travail, le bridage, les outils, le palpeur et son étalonnage, et pour chaque entité : la dimension nominale, la tolérance et son origine, l'estimation du jumeau, la valeur mesurée, l'écart et le verdict. Il dit pourquoi une entité n'a pas été mesurée, et montre la pièce du jumeau vue de dessus.

Pour l'écrire, demandez à l'assistant, par exemple : « Écrivez le rapport du dernier contrôle ». Un même enregistrement donne les mêmes fichiers, octet pour octet.

Où se trouvent les rapports

Les rapports de contrôle et les fiches de bridage se trouvent dans le dossier des rapports :

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

Si vous avez déplacé le dossier des programmes dans les paramètres, le dossier des rapports se trouve à l'intérieur.

Vérifier une copie par rapport au journal d'audit

Studio inscrit dans son journal d'audit le SHA-256 de chaque enregistrement, rapport et fiche de bridage. Chaque entrée est chaînée à la précédente. Vous pouvez vérifier une copie que quelqu'un vous envoie :

- Demandez à l'assistant : « Vérifiez ce fichier par rapport au journal d'audit », avec le chemin du fichier.
- Ou exécutez, sur l'ordinateur où il a été écrit :

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

La réponse indique si le journal contient le SHA-256 du fichier, quand il a été écrit, et si la chaîne du journal est intacte. Une copie modifiée d'un seul bit n'est pas retrouvée.

Fiches de bridage

Une fiche de bridage est une page destinée à l'opérateur devant la machine. Elle dit quoi mettre en place et dans quel ordre. Studio en écrit une pour chaque travail, dans la langue que vous utilisez. Le papier est de format A4, ou Letter aux États-Unis et au Canada.

Pour en écrire une, demandez à l'assistant IA, par exemple : « Écrivez la fiche de bridage de bracket.nc ». L'assistant utilise l'outil `setup_sheet`.

> **Remarque**
>
> Les fiches de bridage sont dans l'édition Pro.

Ce que montre une fiche de bridage

- Programmes : le nom, la version et le nombre de lignes de chaque programme, et les 8 premiers chiffres hexadécimaux de son SHA-256. Comparez-les avec le programme chargé dans la machine.
- Brut et montage : les dimensions et la matière du brut, et la façon dont le jumeau le bride. Pour un étau : les cales parallèles, la prise et le dépassement du brut au-dessus des mors.
- Outils : le numéro, le type, le diamètre, le nombre de dents, la longueur de sortie et la désignation telle qu'écrite de chaque outil, et les opérations qui l'utilisent.
- Chaque bridage : comment la pièce y arrive (retournée, tenue autrement), où se trouve l'origine pièce, et les cycles de palpage qui la trouvent. Les cycles portent les noms qu'ils ont sur le panneau du palpeur.
- Course : les X, Y et Z les plus bas et les plus hauts où le programme se déplace, en coordonnées pièce.
- Dégagement en Z : voir Dégagement en Z.
- Opérations : la stratégie de chaque opération et la façon dont elle entre dans la matière, son outil, sa vitesse, son avance et sa profondeur, et la durée qu'elle prend.
- Vue de dessus : le brut, les mors de l'étau, l'origine pièce et le contour de chaque entité.
- Remarques : les remarques de la FAO, en anglais.

Les durées sont estimées à partir du programme : ses avances, ainsi que la vitesse de rapide et l'accélération de la machine. Les changements d'outil ne sont pas comptés. Un essai dans le jumeau donne des durées où l'accélération et la décélération sont comptées.

Dégagement en Z

Pour chaque bridage, la fiche vérifie que le palpeur et chaque outil passent au-dessus du bridage. Elle vérifie aussi que chaque outil atteint les points les plus hauts et les plus bas du programme. Les hauteurs sont celles du budget en Z du jumeau, mesurées depuis le dessus de la table.

Élément	Doit atteindre
Palpeur	1 mm au-dessus du point le plus haut du bridage, avec Z près du haut de sa course
Chaque outil, en haut	0,5 mm au-dessus du point le plus haut du bridage, et le point le plus haut du programme
Chaque outil, en bas	Le point le plus bas de ses propres opérations

Chaque ligne indique **Suffisant** ou, en rouge, combien de millimètres il manque.

> **Attention**
>
> Une ligne en rouge signifie que le programme ne peut pas s'exécuter sur ce bridage. Utilisez un outil dont la longueur de sortie est moindre ou un palpeur plus court, ou tenez la pièce plus bas, par exemple avec des brides.

Une bride peut dépasser le point le plus haut du programme. La fiche le dit alors : le programme ne doit pas passer au-dessus de la bride. Un essai dans le jumeau le vérifie.

Impression et signature

1. Ouvrez la fiche de bridage dans un navigateur web.
2. Imprimez-la. Résultat : chaque bridage après le premier commence sur une nouvelle page.
3. À la machine, remplissez le cadre de la fin : l'opérateur, la date et le contrôle du premier article (conforme ou non conforme, et le numéro de son rapport de contrôle).

Une nouvelle fiche de bridage du même travail remplace le fichier. Son SHA-256 est inscrit dans le journal d'audit.

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Connecter une machine réelle

Ébauche (itération 14) : le texte est en cours de rédaction ; les sections peuvent encore s'étoffer.

Deux façons de travailler avec une machine

- Pilotez la MC-1 depuis votre propre logiciel d'envoi (bCNC, gSender ou Universal G-code Sender) sur son port USB. Voir Logiciels d'envoi.
- Placez le jumeau à côté, en mode ombre (aperçu) : votre logiciel d'envoi se connecte à Studio au lieu de la machine, Studio transmet chaque ligne à la machine et exécute les mêmes lignes dans le jumeau, puis compare les deux.

L'envoi d'un programme à une machine réelle depuis Studio même ne fait pas partie de la version 1.0.

Mode ombre (aperçu)

> **Remarque**
>
> En 1.0, le mode ombre est un aperçu que l'on démarre depuis la ligne de commande. Il a été testé avec un second jumeau tenant lieu de machine, pas encore avec une machine réelle. La fenêtre n'a pas encore d'endroit pour choisir le port, se connecter ou voir la comparaison ; cela viendra dans une version ultérieure.

Démarrer le mode ombre

Il vous faut : la MC-1 allumée et reliée à cet ordinateur par USB ; votre logiciel d'envoi installé.

1. Trouvez le port de la machine. Sous Windows : **Gestionnaire de périphériques > Ports (COM et LPT)**, par exemple COM5.
2. Démarrez Studio avec ce port depuis une invite de commandes :

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

   Résultat : Studio ouvre le port et le jumeau suit la machine.

3. Dans votre logiciel d'envoi, connectez-vous au port d'envoi de Studio, `localhost:2323`, au lieu de la machine. Résultat : les lignes de votre logiciel d'envoi vont à la machine par l'intermédiaire de Studio, et le jumeau exécute les mêmes lignes.

Le port d'envoi de Studio n'écoute que sur cet ordinateur. Pour l'atteindre depuis un autre ordinateur, ajoutez `--bind any`.

Ce que compare le jumeau

- Les réponses, ligne par ligne : chaque `ok` ou erreur, et les alarmes dans leur ordre.
- La trajectoire : la position que la machine indique par rapport à la trajectoire du jumeau, et de combien le jumeau est en avance ou en retard.
- Les réglages : une fois, quand la machine est connectée et avant que votre logiciel d'envoi se connecte. Connectez votre logiciel d'envoi après que Studio s'est relié à la machine.
- Le contrôleur : une fois, avec les réglages. Si les deux exécutent le même micrologiciel et, avec le noyau propre à Twinwright des deux côtés, sa version et ce que comporte la carte de la machine : broche, codeur, manivelle électronique, axes, palpeur, connexion. Le jumeau ne modélise ni les ports auxiliaires de la carte ni sa carte SD : la comparaison les liste, et ils ne comptent pas comme une différence.
- Les durées d'exécution des parties du programme.

Le garde-fou

Démarrez aussi Studio avec `--shadow-guard`, et le jumeau vérifie chaque ligne de votre logiciel d'envoi avant qu'elle n'atteigne la machine. Quand le jumeau prévoit un crash, il retient la ligne et met la machine en arrêt d'avance. La fenêtre affiche l'état du garde-fou sous **Diagnostic**.

Ce qui ne peut pas atteindre la machine

En mode ombre, seules les lignes de votre logiciel d'envoi atteignent la machine, avec les propres questions du jumeau qui ne changent rien (les réglages et l'état).

- Les commandes de la fenêtre n'agissent que sur le jumeau, et celles qui déplaceraient ou démarreraient la machine sont refusées tant que le jumeau suit une machine. Le panneau du palpeur est verrouillé.
- Un client IA ne peut que lire : il voit l'état de la machine et ne change rien.
- Une manivelle électronique externe connectée à Studio n'agit que sur le jumeau.

> ⚠ **Avertissement**
>
> En mode ombre, l'arrêt et l'arrêt d'urgence de la fenêtre n'arrêtent pas la machine. Pour arrêter la machine, utilisez son propre arrêt d'urgence ou l'arrêt de votre logiciel d'envoi.

Logiciels d'envoi

Les logiciels d'envoi testés avec le noyau propre à Twinwright, et comment les configurer : Logiciels d'envoi.

Quand quelque chose diffère

Une différence signifie que le profil ou les réglages du jumeau ne sont pas ceux de cette machine, ou que la machine fait quelque chose que le modèle ne fait pas. Vérifiez d'abord les réglages : la comparaison liste ceux qui diffèrent.

L’assistant IA

Une IA peut travailler sur le jumeau avec les mêmes outils que vous : elle prépare, écrit et vérifie des programmes et les exécute dans la simulation. Vous pouvez parler à une IA dans l’assistant propre à Studio, ou connecter un client IA tel que Claude Desktop. Tout ce qui modifierait la machine de la fenêtre attend votre accord. L’IA ne fait partie d’aucune fonction de sécurité.

Activer l’IA

L’IA est désactivée dans une nouvelle installation. Pour l’activer, faites l’une de ces choses :

- cliquez sur l’état de l’IA dans la barre du haut (**IA désactivée**) et choisissez **Interrupteur IA de la fenêtre** ;
- **Outils > Paramètres… > IA et MCP** : **Interrupteur IA**, puis **OK** ;
- envoyez un message dans l’assistant : cela active l’interrupteur.

Studio conserve l’interrupteur tel que vous l’avez laissé pour le prochain démarrage. Le menu **IA** (**Assistant IA…**, **Activité IA…**, **Connecter des clients IA…**) apparaît à partir du niveau Opérateur.

Quand l’interrupteur est désactivé, les clients IA peuvent encore utiliser des machines d’entraînement en arrière-plan, mais pas la machine de votre fenêtre.

Le service d’IA et sa clé

L’assistant s’adresse à un service d’IA que vous choisissez, avec votre propre compte :

- **Anthropic (Claude)**, **OpenAI**, **DeepSeek**, **Qwen (Tongyi)**, **Kimi (Moonshot AI)**, **Zhipu (GLM)** ;
- **LM Studio (local)** ou **Ollama (local)** : un modèle sur votre propre ordinateur, sans clé ;
- **Personnalisé (compatible OpenAI)** : tout service doté d’une interface compatible OpenAI, à l’adresse que vous saisissez.

Pour le configurer :

1. Ouvrez **IA > Assistant IA…**
2. Choisissez le **Fournisseur**. Studio renseigne son **URL de base**.
3. Collez votre clé dans **Clé API** et cliquez sur **Enregistrer la clé**. Résultat : la clé est stockée sur cet ordinateur, chiffrée. Sous Windows, seul votre compte Windows peut la lire. Sous Linux, elle va dans le trousseau du bureau (Secret Service : GNOME Keyring, KWallet) ; sans trousseau, elle est chiffrée avec une clé contenue dans un fichier que seul votre compte utilisateur peut lire, et la ligne sous la clé le dit.
4. Cliquez sur **Lister les modèles** et choisissez un **Modèle**.

Studio n’écrit pas la clé dans ses paramètres, dans le registre ni dans le journal d’audit. Pour supprimer une clé enregistrée, videz le champ et cliquez sur **Enregistrer la clé**.

> **Remarque**
>
> Ce que vous envoyez au service d’IA va à ce service, sous votre compte et selon ses conditions : vos messages et, lorsque **Joindre le programme, le bridage et les résultats de vérification de la fenêtre** est activé, le programme, le bridage et les résultats de vérification, ainsi que les résultats des outils que l’IA utilise. Pour que tout reste sur votre ordinateur, utilisez un modèle local.

Avant votre premier message à un service cloud, l’assistant dit une fois ce qui y part. Cliquez sur **Envoyer** pour continuer, ou sur **Utiliser un modèle local…** pour passer à LM Studio ; votre message reste dans la zone de saisie. **Outils > Paramètres… > IA et MCP** liste les services que vous avez acceptés ; **Redemander** oblige l’assistant à redemander.

Le panneau de l’assistant

IA > Assistant IA… ouvre l’assistant dans une fenêtre à part.

- Saisissez votre texte dans la zone en bas. Entrée envoie ; Maj+Entrée passe à une nouvelle ligne. **Arrêter** interrompt la réponse.
- **Joindre le programme, le bridage et les résultats de vérification de la fenêtre (s’ils ont changé)** donne à l’IA, avec chaque message, ce qui se trouve dans votre fenêtre. Cette option est activée par défaut.
- La conversation montre chaque outil que l’IA utilise, avec ✓ quand il a fonctionné et ✗ quand il n’a pas fonctionné.
- Les images que l’IA produit (la pièce, la trajectoire d’outil, l’écart par rapport au modèle, une comparaison) apparaissent dans la conversation. Double-cliquez sur l’une d’elles pour l’agrandir ou l’enregistrer.
- **Nouvelle conversation** repart de zéro. Les conversations ne sont pas enregistrées quand vous fermez Studio.

Fermer la fenêtre ne fait que la masquer ; la conversation reste jusqu’à ce que vous quittiez.

Ce que l’IA peut et ne peut pas faire

Quand l’interrupteur est activé, l’IA peut :

- lire l’état des machines, faire la prise d’origine, déverrouiller, déplacer manuellement et envoyer des lignes de G-code ;
- mettre en place le brut, les outils, l’origine pièce et la pièce, palper et contrôler ;
- vérifier, simuler, optimiser, générer, modifier et charger des programmes, et lancer ou relancer une exécution ;
- lire les exécutions et leurs résultats, les comparer et en faire des images ;

- lire le journal d'audit et l'état de la licence ;
- vous montrer quelque chose dans la fenêtre pendant qu'elle explique : un espace de travail, l'orientation de la vue 3D, la trace, le trajet du programme, la coloration, un onglet du panneau inférieur, une étape de **Préparation** (seul ce que montre la fenêtre change) ;
- chercher dans ce manuel et répondre d'après lui ;
- changer la langue, la région, les unités, la devise et le taux horaire machine, seulement une fois que vous avez cliqué sur **Autoriser** (voir ci-dessous), avec le même effet que dans **Outils > Paramètres…**.

Les outils qui relèvent d'une édition payante (contrôle, rapports, imbrication, optimiseur) ne fonctionnent que dans cette édition.

L'IA ne peut pas modifier les autres paramètres de Studio : le niveau utilisateur, l'interrupteur IA, la manière dont Studio vous interroge, la confidentialité, le journal ou l'apparence ; elle ne peut pas non plus modifier la licence ni le journal d'audit.

Demander avant de modifier la machine

Avant que l'IA modifie la machine de la fenêtre (dans le menu de l'IA de la barre du haut et dans **Outils > Paramètres… > IA et MCP**) propose deux choix :

- **Me demander à chaque fois** (le choix par défaut) ;
- **Ne pas demander (l'IA agit directement sur la fenêtre)**.

Quand l'IA veut modifier la machine de la fenêtre (la déplacer, la mettre sous tension, relâcher l'arrêt d'urgence, charger ou exécuter un programme, modifier le bridage ou les outils) ou l'un des paramètres qu'elle peut modifier, une barre s'ouvre au-dessus de la barre machine. Elle indique quelle IA demande et ce qu'elle veut faire, et affiche un compte à rebours.

1. Lisez la question.
2. Cliquez sur **Autoriser** ou **Refuser**. Entrée ne répond pas : utilisez la souris, ou passez avec Tab sur une touche et appuyez sur Espace.

Sans réponse en 60 s, la demande compte comme **Refuser**. Un accord couvre aussi, pendant deux minutes, les demandes de suite propres à cette action.

Studio ne demande jamais avant que l'IA arrête la machine, la mette en arrêt d'avance ou la réinitialise, appuie sur l'arrêt d'urgence ou se contente de lire. Les machines d'entraînement et les copies cachées sur lesquelles l'IA simule ne donnent lieu à aucune demande.

Lecture seule sur une machine réelle

Quand le jumeau de la fenêtre suit une machine réelle (mode ombre, un aperçu dans la version 1.0 ; voir Une machine réelle), l'IA ne fait que lire et analyser. Elle ne peut pas déplacer la machine, modifier son bridage ni ses décalages d'origine, ni y exécuter des programmes, et elle ne demande pas : Studio refuse ces demandes. Les machines d'entraînement fonctionnent comme avant.

Activité IA

IA > Activité IA… montre ce que font les clients IA sur cet ordinateur :

- **Machines en arrière-plan** : les machines d'entraînement que l'IA utilise, que vous ne voyez pas dans la fenêtre (« window » est la machine de votre fenêtre) ;
- **Exécutions** : les 16 dernières simulations et optimisations ;
- **Clients IA connectés** : qui est connecté, comment, et son dernier appel.

D'ici, vous pouvez arrêter une exécution, regarder l'image d'une exécution, fermer une machine en arrière-plan, et arrêter ou démarrer l'hôte des jumeaux. L'hôte des jumeaux est le programme d'arrière-plan qui contient les machines d'entraînement. Il démarre quand un client IA se connecte et s'arrête au bout de 30 minutes d'inactivité.

L'onglet **Activité IA** du panneau inférieur liste les appels de l'IA sur la machine de la fenêtre : heure, appel, machine, qui et résultat.

Connecter d'autres clients IA (MCP)

Les clients IA qui parlent MCP (Model Context Protocol) peuvent utiliser les outils de Studio.

1. Ouvrez **IA > Connecter des clients IA…**
2. Cliquez sur votre client : **Claude Desktop**, **Application de bureau ChatGPT (modes Codex et Work)**, **Cursor**, **VS Code (GitHub Copilot)** ou **LM Studio (modèles ouverts en local)**. Résultat : pour Claude Desktop et l'application de bureau ChatGPT, Studio montre la modification qu'il va apporter aux paramètres du client et l'effectue quand vous cliquez sur **OK** ; il conserve l'ancien fichier du client à côté. Les autres ouvrent leur propre page d'installation.
3. Quittez complètement le client et rouvrez-le.
4. Activez l'**Interrupteur IA de la fenêtre** dans Studio.

Pour tout autre client, **Autres clients : copier la configuration** copie les paramètres à y coller. Le client lance `TwinwrightStudio.exe --mcp` en arrière-plan ; **La commande et la configuration** montre la commande exacte.

Clients qui n'acceptent qu'une adresse

Pour les clients qui n'acceptent qu'une adresse, activez **URL locale (HTTP, pour les clients qui n'acceptent qu'une URL)** dans le menu de l'IA et utilisez **Copier l'URL locale (HTTP)**. L'adresse ne fonctionne que sur cet ordinateur et demande le jeton qui l'accompagne. **Nouveau jeton** fait cesser de fonctionner l'ancien.

Les outils, leurs arguments et leurs résultats : Outils MCP.

Ce qui est enregistré

Outils > Journal d'audit… (Ctrl+L) enregistre ce que fait l'IA, avec le nom de l'IA dans la colonne **Qui**. Filtrez **Qui** sur **IA** pour ne voir que les enregistrements de l'IA ; double-cliquez sur un appel pour voir tous ses enregistrements ensemble.

Au niveau par défaut **Standard**, le journal conserve :

- l'activation et la désactivation de l'interrupteur IA, et les changements de la manière dont Studio demande ;
- chaque question, votre réponse et chaque action de l'IA sur la machine de la fenêtre ;
- chaque appel d'outil, avec une forme abrégée de ses arguments ;
- la connexion et la déconnexion des clients ;
- les messages de l'assistant : les 180 premiers octets dans le résumé, jusqu'à 2 000 octets dans l'enregistrement.

Les clés, les jetons et les mots de passe sont inscrits sous la forme `***`.

Langue et unités

Au premier démarrage, Studio suit la langue de votre système. Vous pouvez à tout moment changer la langue, les unités et l'aspect des nombres, des dates et des montants. Les programmes et les fichiers restent tels quels.

Changer la langue

Choisissez **Outils > Language / 语言** puis une langue : English, 简体中文 (chinois simplifié), Français, Deutsch, 日本語 ou Español. Le nom du menu est le même dans toutes les langues, de sorte que vous pouvez le trouver dans une langue que vous ne lisez pas.

La fenêtre change aussitôt ; vous n'avez pas à redémarrer Studio. Les lignes déjà présentes dans la console restent dans la langue où elles ont été écrites.

Outils > Paramètres… > Général > Langue offre un choix de plus, **Comme Windows** : Studio suit alors la langue du système à chaque démarrage. Pour une langue dont Studio n'a pas de pack, il revient à l'anglais.

Unités : millimètres et pouces

Outils > Unités permet de choisir **Selon la région**, **Métrique (mm)** ou **Pouce**. Le changement s'applique aussitôt.

En pouces, Studio affiche :

Grandeur	Métrique	Pouce
Longueur	mm	in
Avance	mm/min	in/min
Vitesse de coupe	m/min	ft/min
Force	N	lbf
Couple	N·m	lbf·in
Puissance	W	hp
Température	°C	°F

La vitesse de broche reste en tr/min. Les pas du déplacement manuel (jog) deviennent 0,0005, 0,005, 0,05, 0,5 et 2 in, et les vitesses de déplacement manuel 4, 12, 60 et 120 in/min.

Les unités changent ce que Studio affiche et lit, pas vos fichiers. Un programme s'exécute tel qu'il est écrit, en `G20` ou `G21`. Les programmes que Studio produit sont écrits en millimètres (`G21`). Les outils de l'IA et les journaux travaillent en millimètres.

Région : nombres, dates et montants

Outils > Paramètres… > Général > Région règle l'aspect des nombres, des montants et des dates : **Comme Windows (Paramètres > Heure et langue)**, **Comme la langue**, ou une région de la liste. Sur la même page :

- **Devise** : un code ISO 4217 tel que USD, EUR ou CNY ; vide, c'est celle de la région ;
- **Symbole monétaire** : vide, c'est celui de la devise ;
- **Taux horaire machine** : ce que vous coûte une heure de machine, pour les estimations de coûts ; avec 0, aucune n'est affichée.

La région détermine le séparateur décimal et le séparateur de milliers dans les nombres de Studio, les dates dans les exécutions, les versions et les rapports, et le format de papier des rapports (Letter aux États-Unis et au Canada, A4 ailleurs). Les nombres qui appartiennent à la machine, comme les positions et les nombres d'un programme, utilisent toujours un point et aucun séparateur de milliers.

Saisir des nombres

Studio lit strictement un nombre saisi dans les champs où une erreur coûterait une pièce : les champs de la FAO, les dimensions et les positions dans **Préparation**, les longueurs du palpeur et la vitesse de broche.

- Utilisez un seul séparateur décimal : un point ou une virgule (`13.3` ou `13,3`).
- N'écrivez pas de séparateur de milliers.
- Vous pouvez écrire l'unité du champ après le nombre : `13 mm`, `0.5 in`, `30°`.
- Les chiffres saisis avec une méthode de saisie (chiffres pleine chasse) sont lus aussi.

`1,000` pose une question quand votre région écrit le séparateur décimal sous forme de point : voulez-vous dire un ou mille ? Studio ne devine pas ; il vous demande d'écrire celui que vous voulez dire. Là où la région écrit une virgule, `1,000` vaut un.

Ce que signifie « lu comme »

Quand ce que vous avez saisi n'est pas écrit comme Studio écrit les nombres, le champ montre ce que Studio a lu : saisissez `13,3` et il affiche **lu comme 13.3**. En pouces, il affiche quatre décimales : **lu comme 0.3750**. Vérifiez-le avant de continuer.

Quand Studio ne peut pas lire ce que vous avez saisi, le champ devient rouge et une ligne dessous en donne la raison, par exemple : « 1,2,3 » n'est pas une longueur. Studio ne retient pas la valeur : le champ garde la dernière qu'il a retenue jusqu'à ce que vous la corrigiez.

Fractions de pouce

En pouces, les champs de longueur acceptent aussi les fractions :

- `3/8`, `5/4`
- `1 1/2` ou `1-1/2`
- `2 3/8"`, `1/2 in` (la marque du pouce peut être `"`, `”` ou `in`)

Le champ montre le décimal et la fraction qu'il a lus : **lu comme 0.3750 (3/8")**. Studio n'accepte pas un point décimal avec une fraction (`1.5/2`), une fraction impropre après un nombre entier (`1 5/4`) ni des espaces autour de la barre oblique.

Studio affiche aussi les dimensions en pouces sous forme de fractions, jusqu'à 1/64 in : les diamètres et les longueurs d'outil, les dimensions du brut, les diamètres et les profondeurs de trou s'affichent `1/4"` ou `1 1/2"`. Une dimension qui n'est pas un multiple de 1/64 s'affiche avec quatre décimales, `0.2010 in`. Les coordonnées, les avances, les vitesses et les mesures sont toujours des décimaux.

En millimètres, une fraction n'est pas lue : « Les champs en millimètres n'acceptent pas les fractions : écrivez 4,7625 ou passez en pouces ».

Les descriptions d'outils lisent les dimensions en pouces dans l'un et l'autre système, par exemple `flat 1/4in n2`.

Vitesse de broche

Vous pouvez saisir la vitesse de broche au lieu de la faire varier par pas de 1 000 tr/min :

1. Dans **Exécution**, ouvrez **Mode manuel** dans la colonne de droite.
2. Double-cliquez sur l'affichage **Vitesse de broche**, ou appuyez sur Entrée dessus.
3. Saisissez la vitesse et appuyez sur Entrée. Échap, ou un clic ailleurs, conserve l'ancienne vitesse.

Quand la broche tourne en mode manuel (aucun programme en cours), la nouvelle vitesse est envoyée aussitôt à la broche. Quand elle est arrêtée, Studio garde la vitesse pour le prochain **Broche sens horaire** ou **Broche sens antihoraire**. Pendant l'exécution d'un programme, c'est le programme qui fixe la vitesse et le champ ne s'ouvre pas.

Une vitesse hors de la plage de la broche rend le champ rouge et n'est pas retenue. La plage du MC-1 est de 1 000 à 24 000 tr/min.

Ce qui reste en anglais

- le G-code, et les fichiers que Studio écrit et lit (travaux, bridages, enregistrements, fichiers de licence) ;
- les outils de l'IA et ce qui va vers les clients IA et en vient ;
- les chapitres de référence produits à partir du programme lui-même : Outils MCP, Ligne de commande et Schéma du profil de machine ;
- le message de chaque enregistrement d'audit relatif aux programmes (le journal le garde aussi dans votre langue) ;
- en allemand, en français, en japonais et en espagnol dans la version 1.0 : les descriptions des erreurs et des alarmes du contrôleur, l'aide des paramètres du contrôleur et certains messages de licence.

Le PDF d'un rapport de contrôle est dans votre langue quand ses polices contiennent les lettres de votre langue, et en anglais sinon ; son fichier CSV est toujours dans votre langue.

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Clavier et accessibilité

La machine, les menus, les touches des panneaux et l'éditeur se commandent tous au clavier. Faire tourner et zoomer la vue 3D, y choisir un point et redimensionner les panneaux demandent une souris. Sous Windows, un lecteur d'écran lit les touches et les affichages des panneaux.

Aide > Raccourcis clavier (ou **?**) liste toutes les touches de Studio.

Les touches de la machine

Touche	Action
F1	**Urgence** : appuyez ; appuyez de nouveau pour relâcher
F2	**Mise sous tension**
F3	**Origine**
F4	**Déverrouiller**
F5	**Départ cycle**
F6 ou Espace	**Arrêt d'avance**
F7	**Arrêt** : arrêt d'avance, puis une réinitialisation une fois la machine à l'arrêt ; appuyez de nouveau pour réinitialiser tout de suite
F8	**Refroidissement** activé ou désactivé
F9	**Broche sens horaire**
F10	**Broche sens antihoraire**
F11	**Arrêt broche**
F12 ou Ctrl+X	**Reset**

Déplacement manuel (jog) et manivelle :

Touche	Action
← →	Déplacement manuel en X
↓ ↑	Déplacement manuel en Y
Pg suiv., Pg préc.	Déplacement manuel en Z
Maj avec l'une de ces touches	Déplacement manuel tant que vous maintenez la touche
1–5	Pas de 0,01, 0,1, 1, 10 ou 50 mm
6	**Maintenir pour déplacer** : chaque touche de déplacement manuel déplace tant que vous la maintenez
Ctrl+1 à Ctrl+4	Vitesse de déplacement manuel de 100, 300, 1 500 ou 3 000 mm/min
Échap	**Arrêter le déplacement manuel**
X, Y, Z	Choisir l'axe de la manivelle
, et .	Tourner la manivelle d'un cran vers le bas ou vers le haut (avec Maj : dix)

Corrections :

Touche	Action
Ctrl+↑, Ctrl+↓	Avance +10 %, −10 %
Ctrl+→, Ctrl+←	Broche +10 %, −10 %
Ctrl+0	Avance et broche remises à 100 %

Chaque appui sur une touche de déplacement manuel fait avancer d'un pas. Un déplacement maintenu s'arrête quand vous relâchez la touche ou quand la fenêtre perd le focus. En pouces, les pas vont de 0,0005 à 2 in et les vitesses de 4 à 120 in/min.

Les touches F fonctionnent où que vous soyez. Les autres touches de la machine ne fonctionnent pas pendant que vous saisissez du texte dans une zone de texte, une liste ou l'éditeur : cliquez sur la vue 3D pour les rendre à la machine.

Les touches des chiffres et les touches , et . sont reconnues d'après leur emplacement sur le clavier : elles marchent avec n'importe quelle disposition de clavier.

Arrêt d'urgence

Appuyez sur F1, ou cliquez sur la touche rouge ronde à l'extrémité droite de la barre machine. Dans le jumeau, cela ouvre le circuit d'arrêt d'urgence comme le fait le bouton de la machine : les variateurs et celui de la broche perdent leur alimentation, et le contrôleur déclenche ALARM:10. Appuyez de nouveau sur F1 pour le relâcher, puis suivez les étapes pour reprendre le travail : **Mise sous tension**, **Reset**, **Origine**.

> ⚠ **Avertissement**
>
> F1 et la touche rouge arrêtent le jumeau. Elles ne constituent pas un arrêt d'urgence pour une machine réelle. Sur une machine réelle, utilisez le bouton d'arrêt d'urgence de la machine elle-même.

Les touches de la fenêtre

Touche	Action
Alt+1 à Alt+6	Aller à **Préparation**, **FAO**, **Programme**, **Exécution**, **Analyse** ou **Diagnostic**
Ctrl+Tab, Ctrl+Maj+Tab	L'espace de travail suivant ou précédent
Ctrl+J	Replier ou déplier le panneau inférieur
Ctrl+Maj+M	Changer de machine
Ctrl+O	**Ouvrir un programme…**
Ctrl+P	**Imprimer le programme…**
Ctrl+,	**Paramètres…**
Ctrl+L	**Journal d'audit…**
Ctrl+Q	**Quitter**
? ou /	**Raccourcis clavier**
Alt et la lettre d'un menu	Ouvrir ce menu ; Alt seul met en évidence les titres des menus
Tab, Maj+Tab	Passer d'une touche à l'autre d'un panneau ; Espace appuie sur la touche qui porte le cadre
Échap	Fermer une fenêtre contextuelle ou une sous-fenêtre

> **Remarque**
>
> F1 est l'arrêt d'urgence, pas l'aide. Pour ouvrir le manuel, utilisez **Aide > Documentation** ou la touche **?** d'un panneau.

L'éditeur de l'espace de travail **Programme** a ses propres touches :

Touche	Action
Ctrl+E	Entrer dans l'éditeur et revenir à la machine
Ctrl+K	**Charger et vérifier**
Ctrl+S	**Enregistrer une version**
Ctrl+Maj+S	Enregistrer le programme dans un fichier
Ctrl+D	**Comparer** avec une version enregistrée
Ctrl+F, Ctrl+G, Ctrl+H	Rechercher, aller à une ligne, remplacer
Ctrl+B	**Point d'arrêt** sur la ligne
Ctrl+Maj+B	Mettre en pause sur une condition (**Pause si**)
Ctrl+.	Appliquer la suggestion de la vérification pour la ligne
Alt+↓, Alt+↑	Aller au problème ou à la différence suivante ou précédente
Maj+F10 ou la touche Menu	Le menu de la ligne
Ctrl+Maj+O	**Optimiser**
Ctrl+Maj+G	**Générer…**
Ctrl+Maj+R	Reprendre le programme qui se trouve dans la machine
Échap	Fermer la comparaison

La fenêtre du manuel : Ctrl+F recherche, Alt+← et Alt+→ reculent et avancent, Échap la ferme.

Pendant que vous disposez une pièce dans **Préparation** : les flèches la déplacent (Maj plus fin, Ctrl plus gros), Q et E la tournent, Entrée la place, Échap la remet.

La liste complète

Aide > Raccourcis clavier montre toutes les touches ci-dessus en une seule liste. Ces commandes n'ont pas de touche ; utilisez leurs menus :

- **Bloc à bloc** et **Bloc suivant** (le menu **Programme** et la barre machine) ;
- **Vérifier le programme** ;
- la mise à zéro des axes et le déplacement vers le zéro (**Machine > Origine pièce**) ;

- les mouvements du capteur de longueur d'outil et du palpeur (**Machine > Capteur de longueur d'outil et palpeur**) ;
- la vitesse de broche par pas de 1 000 tr/min, et la correction de rapide (**Machine**) ;
- la mise en pause de la simulation et sa vitesse (**Simulation**).

Utiliser un lecteur d'écran

Sous Windows, Studio fonctionne avec les lecteurs d'écran par l'interface d'accessibilité du système. Dans chacun des panneaux de Studio, le lecteur reçoit :

- le nom de chaque touche, son type, si elle est activée, ouverte ou indisponible, son raccourci et son infobulle ;
- les affichages sous forme de texte : la barre machine (état, programme, progression, étape suivante, horloge), la barre du haut (machine, état, IA, niveau, édition) et la colonne de droite (position, broche, jauges, ligne modale) ;
- chaque déplacement du focus avec Tab ;
- les menus : le libellé de chaque élément, son raccourci, s'il est indisponible ou coché, et celui où vous vous trouvez.

Deux choses sont annoncées au moment où elles se produisent, où que soit le focus : une alarme (avec l'étape suivante) et la demande d'une IA de modifier la machine.

La fenêtre du manuel donne au lecteur le texte de la page, avec ses titres et ses liens.

Dans la version 1.0 :

- les autres changements d'état de la machine ne sont pas annoncés : placez le focus sur la barre machine pour les entendre ;
- le listing, la console, les messages, l'éditeur et la table d'outils sont lus selon les règles propres au lecteur ; les courbes ne sont pas lues ;
- sous Linux, les lecteurs d'écran ne sont pas pris en charge.

Ce qui est lu dans la vue 3D

La vue 3D elle-même ne donne rien au lecteur. Les nombres qu'elle montre figurent dans la colonne de droite et la barre machine, que le lecteur lit. Quand vous cliquez sur un point de la pièce, la phrase sous la vue (la ligne du programme et la position) est lue.

Grand texte et mode atelier

Studio suit l'échelle d'affichage que vous réglez dans Windows, écran par écran. Il n'a pas de réglage de taille du texte qui lui soit propre.

Affichage > Apparence > Mode atelier (grand format) agrandit la barre machine, ses touches, l'arrêt d'urgence, le texte, le listing et la position, et donne à l'espace de travail **Exécution** une disposition pour lire à bout de bras (voir Clair, sombre et mode atelier).

Quand les animations du système sont désactivées, la vue 3D passe directement à une nouvelle vue au lieu d'y glisser.

Dépannage

Que faire quand quelque chose ne fonctionne pas comme vous l'attendez. Chaque section dit ce que vous voyez, ce que cela signifie d'habitude et où se trouve la réponse complète. Si aucune ne convient, collectez les informations de diagnostic et écrivez-nous (la dernière section).

Studio ne démarre pas

- Studio demande Windows 10 ou 11 en 64 bits ; l'installateur s'arrête sur les autres versions. Voir Configuration requise.
- Si la fenêtre s'ouvre mais que la vue 3D reste vide, ou indique qu'elle n'est pas disponible : **Outils > Paramètres… > Fichiers**, cochez **Rendu logiciel (si la carte graphique pose problème ; au prochain démarrage)**, puis relancez Studio. Mettre à jour le pilote de la carte graphique en corrige souvent la cause. Voir Quand la vue 3D ne peut pas démarrer.
- Si Studio s'est fermé pendant son démarrage, le démarrage suivant propose le rapport (voir Studio s'est fermé de manière inattendue).

La machine est en alarme

En cas d'alarme, l'état sur la barre machine devient rouge. La barre nomme l'alarme et ce qu'elle signifie, et donne l'**Étape suivante** avec la touche qui l'exécute ; **Toutes les étapes** liste le chemin pour reprendre le travail quand il y a plus d'une étape. Cliquez sur la touche, ou utilisez **Machine > Effectuer l'étape suivante** : Studio recalcule les étapes après chacune.

- Un mouvement de palpage qui n'a rien touché s'arrête avec ALARM:5 : effectuez les étapes (**Reset**, **Déverrouiller**), puis vérifiez où le cycle attendait la face.
- Un mouvement au-delà de la course de la machine s'arrête à la limite logicielle : regardez la ligne que le listing marque en rouge et l'origine pièce avec laquelle il s'est exécuté.
- Après une alarme, la position peut ne plus être fiable (une ligne sous **Position** l'indique) : refaites la prise d'origine avant d'usiner.

Les étapes pour chaque cas : Alarmes et conduite à tenir. Chaque alarme, ses causes et ce qui la lève : Alarmes.

Une ligne du programme répond par une erreur

`error:` suivi d'un nombre signifie que le contrôleur a refusé cette ligne : il ne l'a pas exécutée, et la machine ne s'est pas déplacée pour elle. Le message donne le numéro et ce qu'il signifie.

- **Programme > Vérifier le programme** fait passer tout le programme par le mode de vérification du contrôleur sans que la machine bouge, et liste les premières lignes qu'il a refusées : « ligne 5 : error:22 (…) ». Corrigez ces lignes et vérifiez de nouveau.
- Les causes les plus courantes sont un code que le contrôleur n'accepte pas, un mot manquant (un arc sans son centre ni son rayon, un mouvement de coupe avant tout `F`) ou deux codes dans une même ligne qui ne peuvent pas aller ensemble.

Chaque erreur et ses causes : Erreurs.

La vérification du programme trouve un problème

L'éditeur vérifie le programme pendant que vous saisissez, au chargement et après une exécution. Chaque problème trouvé s'affiche sous sa ligne avec le numéro de sa règle et son type : une erreur, un avertissement ou une remarque.

- Une **erreur** est une ligne que le contrôleur refuserait ou un mouvement que le bridage ne permet pas ; corrigez-la avant d'exécuter.
- Un **avertissement** est une ligne qui s'exécute mais peut ne pas faire ce que vous voulez : un rapide sous le dessus de la pièce, une broche lancée sans `S`, un changement d'outil après lequel le programme descend avant d'être au-dessus de son point de départ.
- La partie de la vérification qui fait l'aperçu demande que la machine soit sous tension et que la prise d'origine soit faite, sans programme en cours ; d'ici là, elle attend.

Les règles et la façon de corriger un problème : La vérification du programme. Ce qu'une exécution dans le jumeau vérifie en plus : Ce que vérifie le jumeau.

Le cycle de palpage ne démarre pas

Démarrer le palpage… reste grisé tant qu'une vérification de la carte du palpeur est rouge, ou tant qu'il manque quelque chose ; la ligne sous les touches dit alors quoi (indiquez X et Y, indiquez le diamètre de la bague, attendez la fin d'un appel en cours).

- **La broche ne porte pas de palpeur** : dans **Préparation**, étape **Outil**, choisissez le palpeur et cliquez sur **Changer d'outil**.
- **La broche tourne** : arrêtez-la (**Arrêt broche**, F11).
- **Pas de place au-dessus du brut pour passer** : la ligne donne la hauteur de la pointe du palpeur avec Z en haut, le haut du bridage et ce qui manque. Utilisez un palpeur plus court, ou tenez le brut plus bas : des brides ou du ruban adhésif double face au lieu de l'étau.
- **Le jumeau suit une machine réelle** : palpez-y avec la commande propre à la machine.
- Un cycle qui n'a rien touché s'arrête avec ALARM:5 (voir La machine est en alarme).

Voir Quand un cycle s'arrête.

Licence et activation

Aide > Licence montre l'édition en vigueur, l'état de la licence et sa date de fin.

- **Activer en ligne** : saisissez le code tel que vous l'avez reçu ; **Activer** devient disponible dès que le code se lit comme tel. La réponse du serveur s'affiche dessous. Quand la réponse indique que la licence est sur trop d'ordinateurs, qu'elle est terminée, en pause, impayée ou remboursée, **Ouvrir la page des licences** l'ouvre sur notre site web : vous y libérez un ordinateur, renouvelez ou gérez l'abonnement.
- **Sans internet** : la même boîte de dialogue contient les étapes hors ligne. Studio donne un code de demande ; sur un ordinateur connecté à internet, vous obtenez le fichier de licence correspondant sur la page des licences, puis vous importez ce fichier ici.
- Un abonnement qui n'est pas renouvelé continue de fonctionner 14 jours de plus (la licence affiche **En période de grâce**). Passé ce délai, ou quand une licence prend fin, Studio redevient l'édition Learning ; vos projets et vos programmes restent intacts.

Voir Licences et Quand une licence prend fin.

L'assistant IA ne répond pas

- La ligne sous la zone de saisie dit ce que fait l'assistant (attente du service, jumeau en train d'exécuter un outil).
- Vérifiez le **Fournisseur**, l'**URL de base** et le **Modèle** ; pour un service cloud, qu'une clé est enregistrée (**Enregistrer la clé**).
- Pour un modèle local, son serveur doit tourner (le serveur de LM Studio, Ollama) et un modèle doit être choisi (**Lister les modèles**).
- Une réponse peut prendre du temps ; **Arrêter** y met fin.
- Le premier message à un service cloud vous attend : Studio montre une fois ce qui y part, et n'envoie qu'avec votre accord (demandé une fois pour chaque service). **Utiliser un modèle local…** garde tout sur cet ordinateur. Voir Confidentialité.
- Quand l'IA veut modifier la machine de la fenêtre, une barre en bas de la fenêtre vous interroge. Sans réponse en 60 s, la demande compte comme refusée.

Voir L'assistant IA.

Studio s'est fermé de manière inattendue

Au démarrage suivant, Studio demande ce qu'il faut faire du rapport de ce qui s'est passé : ouvrir son dossier, ou l'enregistrer dans un fichier. Rien n'est envoyé. Le rapport laisse de côté les programmes, les noms de fichiers et de dossiers, les clés, les codes et les noms de personnes ; les rapports de plus de 30 jours sont supprimés au démarrage. Voir Confidentialité.

Obtenir de l'aide

Aide > Signaler un problème… envoie un problème, une question d'utilisation ou une idée à l'assistance de Twinwright. Dites ce que vous avez fait, ce que vous attendiez et ce que vous avez vu ; donnez une adresse e-mail si vous voulez aussi y recevoir les réponses. Toujours envoyés avec le signalement, et montrés dans la boîte de dialogue (**Tout voir…** les montre en entier) : la version et la compilation, l'édition, le système, sa langue et son écran, et les 50 dernières entrées du journal d'audit en bref – ce qui a été fait et comment cela s'est terminé, sans le contenu des programmes, votre dossier personnel écrit `~`. **Joindre le paquet de diagnostic** et **Joindre une capture de la fenêtre** ne sont pas cochés à l'ouverture de la boîte de dialogue ; rien n'en part si vous ne les cochez pas. Après **Envoyer**, la boîte de dialogue affiche le numéro du ticket (par exemple `TW-1042`). Sans connexion, le signalement est enregistré dans un fichier, `problem-<time>.zip`, dans `Documents\Twinwright Studio\Reports` : renvoyez-le depuis la boîte de dialogue, ou téléversez-le sur la page Assistance du site web. **Enregistrer dans un fichier…** l'enregistre sans l'envoyer.

Un client IA peut rédiger un signalement pour vous (report_draft) : la même boîte de dialogue s'ouvre avec son texte et une ligne qui indique quel client l'a écrit. Relisez-le, modifiez ce que vous voulez ; vous seul pouvez l'envoyer, et vous seul pouvez joindre le paquet ou la capture.

Aide > Mes signalements liste les signalements envoyés depuis cet ordinateur : le numéro de chacun, son état (reçu, en cours, répondu, résolu) et la date de son dernier changement ; un point marque une réponse que vous n'avez pas lue. Choisissez-en un pour lire l'échange ; **Ajouter un complément…** envoie autre chose au même ticket (**Joindre le paquet de diagnostic…** y ajoute le paquet). Les réponses se trouvent aussi dans « Mes tickets » de l'espace client. Tant qu'un signalement n'est pas résolu, Studio vérifie toutes les 30 minutes s'il y a de nouvelles réponses et le menu Aide affiche combien sont nouvelles ; **Paramètres > Général > Vérifier toutes les 30 minutes les réponses à mes signalements** désactive cela. Ce qui est envoyé et conservé : Confidentialité.

Aide > Collecter les informations de diagnostic… place ce dont le support a besoin dans un seul fichier ZIP dans `Documents\Twinwright Studio\Reports` : la version, les paramètres (clés écrites `***`), l'archive de la machine, les paramètres du contrôleur, les 7 derniers jours du journal d'audit, les rapports de plantage et l'état de la licence. **Le programme et le bridage actuels** est omis sauf si vous le cochez. Les chemins, les noms de fichiers et le nom d'utilisateur de l'ordinateur sont supprimés.

Ouvrez le fichier et regardez-le avant de l'envoyer ; ce fichier ne part pas tout seul (un signalement ne le prend que si vous cochez **Joindre le paquet de diagnostic**). L'adresse à laquelle écrire se trouve dans **Aide > À propos**.

Operations manual

Configuration requise

Ce dont un ordinateur a besoin pour faire fonctionner Twinwright Studio, et ce que la vue 3D attend de sa carte graphique.

Windows

- Windows 10 ou Windows 11 en 64 bits (x64). Le programme d'installation s'arrête sur les autres versions.
- Rien d'autre à installer : Studio tient en un seul fichier programme et n'utilise que les bibliothèques fournies avec Windows.

Linux

- Linux sur x86-64 avec glibc 2.38 ou plus récent : Ubuntu 24.04, Debian 13, Fedora 39 et suivants, ainsi que les distributions qui en dérivent.
- Un bureau avec X11 ou Wayland.
- Pour l'AppImage : FUSE 2 (sous Ubuntu, le paquet `libfuse2t64`). Sans lui, prenez le `.tar.gz`.
- Pour l'activation et les essais en ligne : libcurl (sous Ubuntu, le paquet `libcurl4t64` ; les bureaux l'ont en général). Sans elle, Studio s'active hors ligne.
- Pour les clés des services d'IA : le trousseau du bureau (Secret Service : GNOME Keyring ou KWallet). Sans trousseau, Studio garde les clés dans un fichier que vous seul pouvez lire.

Carte graphique

La vue 3D utilise Direct3D 11, sur toute carte graphique ou tout processeur graphique intégré qui le prend en charge (niveau de fonctionnalités 10.0 ou supérieur). À défaut, Studio dessine avec le moteur de rendu logiciel de Windows (WARP) : cela fonctionne, mais la vue est plus lente (30 images par seconde au plus) et il n'y a pas de gros plan sur l'outil.

Aide > À propos indique, sous **Affichage**, lequel des deux Studio utilise.

Quand la vue 3D ne peut pas démarrer

Le reste de la fenêtre fonctionne sans la vue 3D. Si la vue reste vide ou indique qu'elle n'est pas disponible :

1. Ouvrez **Outils > Paramètres… > Fichiers**.
2. Cochez **Rendu logiciel (si la carte graphique pose problème ; au prochain démarrage)**.
3. Fermez Studio, puis relancez-le.

Mettre à jour le pilote de la carte graphique en élimine souvent la cause.

Linux

Sous Linux, la vue 3D utilise OpenGL 4.1 ou plus : les pilotes Mesa l'ont pour les cartes graphiques des dix dernières années. Sans pilote graphique, le moteur de rendu logiciel de Mesa dessine la vue, plus lentement ; pour l'essayer, lancez Studio depuis un terminal avec `LIBGL_ALWAYS_SOFTWARE=1` devant le nom du programme.

Mémoire et disque

- Mémoire : 8 Go ou plus.
- Disque : environ 62 Mo pour le programme et son manuel, et de la place pour vos données. Vos données augmentent avec l'usage : par défaut, le journal d'audit est conservé 180 jours et dans la limite de 2 Go (**Outils > Paramètres… > Journal**), et chaque version de programme et chaque rapport sont conservés (voir Vos données et sauvegardes).

Écrans et mise à l'échelle

- Au moins 1280 × 720 ; Studio est conçu pour 1600 × 1000 et plus.
- Toute échelle d'affichage. Studio suit l'échelle de chaque écran, et déplacer la fenêtre vers un autre écran l'adapte à la nouvelle échelle.

Une machine réelle

En 1.0, la connexion à une machine réelle est un aperçu, en ligne de commande (voir Une machine réelle) : un port série (USB) à 115 200 bauds, ou un contrôleur sur le réseau.

Studio utilise le réseau comme suit :

- les logiciels d'envoi de G-code se connectent au jumeau sur cet ordinateur, port 2323 ;
- la vue dans le navigateur utilise le port 8765, sur cet ordinateur ;
- les clients IA qui acceptent une adresse utilisent le port 8766, sur cet ordinateur, lorsque vous l'activez ;
- l'activation et le renouvellement en ligne passent par le serveur de licences.

Les ports de Studio n'écoutent que sur cet ordinateur : les autres ordinateurs ne peuvent pas les atteindre.

Sous Linux, le port série appartient au groupe `dialout` : ajoutez-y votre compte une fois (`sudo usermod -aG dialout $USER`) puis reconnectez-vous.

Installer et désinstaller

Comment installer Twinwright Studio, y compris sans questions sur de nombreux ordinateurs à la fois, ce qui va où, et comment le désinstaller.

Pour Linux, voir Sous Linux.

Installer

1. Lancez `TwinwrightStudio-<version>-setup.exe`.
2. Lisez le contrat de licence et cliquez sur **I Agree**.
3. Gardez le dossier (`%LOCALAPPDATA%\Programs\Twinwright Studio`) ou choisissez-en un autre.
4. Sur la dernière page, choisissez si Studio doit démarrer et si un raccourci doit être placé sur le bureau.

Le programme d'installation s'affiche en anglais ou en chinois, selon la langue du système.

Pour vous ou pour tous les utilisateurs de l'ordinateur

Le programme d'installation installe Studio pour la personne qui le lance. Il n'exige aucun droit d'administrateur, et l'entrée du menu Démarrer et le raccourci du bureau sont ceux de cette personne. Chaque personne qui utilise l'ordinateur installe Studio pour elle-même.

La licence appartient à l'ordinateur, pas à la personne : tous les utilisateurs de l'ordinateur la partagent (voir Le dossier des licences).

Studio ne modifie aucune association de fichiers, ni le `PATH`, ni les paramètres du pare-feu, et n'installe aucun service.

Installer sans questions

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

- `/S` installe sans afficher de pages et ne lance pas Studio.
- `/DESKTOP` ajoute le raccourci du bureau.
- `/D=` fixe le dossier. Cet argument doit être le dernier, sans guillemets, même si le dossier contient des espaces.

Pour désinstaller Studio sans questions : `uninstall.exe /S` dans le dossier du programme.

Ce qui va où

Élément	Emplacement
Le programme	`%LOCALAPPDATA%\Programs\Twinwright Studio`
Votre travail : programmes, versions, rapports, journal d'audit	`Documents\Twinwright Studio`
L'état propre du programme : paramètres, machines, clés d'IA	`%LOCALAPPDATA%\Twinwright\Studio`
La licence	`%ProgramData%\Twinwright\licence`

Le dossier du programme

`TwinwrightStudio.exe`, le manuel (`manual\`), `README.txt`, `THIRD-PARTY-NOTICES.txt` et `uninstall.exe`. Le dossier reste le même lors des mises à jour, de sorte que les clients IA que vous avez connectés continuent de fonctionner.

Vos documents

`Documents\Twinwright Studio` (le dossier Documents que Windows vous attribue, y compris lorsqu'il est redirigé) :

- `programs\` : les versions de chaque programme et les versions des travaux FAO ;
- `Reports\` : rapports de contrôle, fiches de bridage et archives de diagnostic ;
- `logs\` : le journal d'audit.

Outils > Paramètres… > Fichiers et **Journal** permettent de placer les programmes et le journal ailleurs (au prochain démarrage).

L'état propre du programme

`%LOCALAPPDATA%\Twinwright\Studio` : les paramètres, la position de la fenêtre, le monde de chaque machine (sa pièce, ses outils, ses décalages d'origine et les réglages de son contrôleur), les clés des services d'IA (chiffrées pour votre compte Windows) et les rapports de plantage (conservés 30 jours).

Le dossier des licences

`%ProgramData%\Twinwright\licence` contient les fichiers de licence, essais compris, pour tous les utilisateurs de l'ordinateur. Studio le crée lors de la première activation ou du premier essai. Si Studio ne peut pas y écrire, un administrateur peut créer le dossier et autoriser les utilisateurs à y écrire.

Pour des configurations particulières, ces variables d'environnement déplacent les dossiers :

Variable	Déplace
`TWINWRIGHT_STATE_DIR`	L'état propre du programme

Variable	Déplace
`TWINWRIGHT_WORKDIR`	Le dossier des programmes
`TWINWRIGHT_LOGDIR`	Le journal d'audit
`TWINWRIGHT_LICENCE_DIR`	Le dossier des licences

Mettre à jour vers une nouvelle version

Fermez Studio, puis lancez le programme d'installation de la nouvelle version. Il s'installe dans le même dossier ; votre travail, vos paramètres, vos machines et votre licence restent tels quels. Studio ne cherche pas lui-même les mises à jour (voir Mises à jour).

Si une copie de Studio tourne encore en arrière-plan (un client IA peut en avoir lancé une), le programme d'installation met l'ancien fichier du programme de côté et le supprime lors de la prochaine mise à jour ou de la prochaine désinstallation.

Désinstaller

Dans Windows, utilisez **Paramètres > Applications > Twinwright Studio > Désinstaller**, ou lancez `uninstall.exe` dans le dossier du programme.

Ce que la désinstallation laisse en place

La désinstallation supprime le programme, ses raccourcis et son entrée dans **Applications**. Elle laisse :

- votre travail dans `Documents\Twinwright Studio` ;
- l'état propre du programme dans `%LOCALAPPDATA%\Twinwright\Studio` ;
- le dossier des licences ;
- l'entrée `twinwright` dans les paramètres des clients IA que vous avez connectés.

Supprimez-les à la main pour tout effacer. Supprimer le dossier des licences ne redonne pas d'essai : le serveur de licences accorde à chaque ordinateur un essai de chaque édition.

Sous Linux

Installer

Studio pour Linux est un seul fichier, une AppImage. Rien n'est installé dans le système.

1. Vérifiez le fichier : `sha256sum TwinwrightStudio-<version>-x86_64.AppImage` doit donner le SHA-256 de la page de téléchargement.
2. Placez-le à un endroit où il peut rester, sous un nom sans version, pour que les clients d'IA connectés continuent de fonctionner après une mise à jour, et rendez-le exécutable :

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

1. Lancez-le : `~/Applications/TwinwrightStudio.AppImage`, ou double-cliquez dessus dans le gestionnaire de fichiers.

L'AppImage a besoin de FUSE 2. S'il ne démarre pas faute de FUSE 2, installez-le (`sudo apt install libfuse2t64` sous Ubuntu 24.04 et suivants), ou prenez plutôt le `.tar.gz` : décompressez-le n'importe où et lancez `TwinwrightStudio` dans son dossier.

Où va quoi sous Linux

Quoi	Où
Le programme, avec son manuel	l'AppImage, là où vous l'avez mise
Votre travail : programmes, versions, rapports, journal d'audit	`~/Documents/Twinwright Studio`
L'état propre du programme : réglages, machines	`~/.local/state/twinwright/studio` (sous `$XDG_STATE_HOME` s'il est défini)
Les clés des services d'IA	le trousseau du bureau ; sans trousseau, un fichier du dossier d'état que vous seul pouvez lire
La licence	`/etc/twinwright/licence`, ou `~/.local/share/Twinwright/licence`

Le dossier de licence est celui de l'ordinateur, pour tous les utilisateurs, quand votre compte peut le créer ou y écrire. Sinon, Studio garde la licence dans `~/.local/share/Twinwright/licence` (sous `$XDG_DATA_HOME` s'il est défini) : pour vous seul, et toujours pour cet ordinateur uniquement. Pour partager une licence entre les utilisateurs d'un ordinateur, créez le dossier une fois avec `sudo install -d -m 1777 /etc/twinwright/licence`.

Essai et activation sous Linux

Les essais, l'activation et le renouvellement passent par internet comme sous Windows, avec la libcurl du système. Sans libcurl ou sans internet, activez hors ligne (voir Licences).

Mettre à jour et désinstaller sous Linux

Pour mettre à jour, quittez Studio et mettez la nouvelle AppImage à la place de l'ancienne, sous le même nom (étape 2 ci-dessus). Pour désinstaller Studio, supprimez l'AppImage. Dans les deux cas restent vos documents, le dossier d'état, le dossier de licence, les clés de votre trousseau et l'entrée `twinwright` dans les clients d'IA connectés ; supprimez-les à la main pour tout enlever.

Licences, postes et activation

Sans licence, Studio fonctionne comme l'édition Learning. Une licence payante ajoute les fonctions de son édition. Ce chapitre s'adresse à la personne qui achète, active et gère les licences. Les prix figurent sur le site web.

Si les licences payantes ne sont pas encore en vente au moment où vous lisez ceci, la page des tarifs du site web l'indique. D'ici là, Studio fonctionne comme l'édition Learning, et vous pouvez essayer Maker ou Pro gratuitement pendant 30 jours (voir Essayer une édition payante) ; les sections sur l'activation s'appliquent dès que les licences sont en vente.

Éditions

Édition	Ajoute	Qui peut l'utiliser
Learning	Tout le jumeau, l'éditeur, la machine réelle, les exemples, la connexion à l'IA, et la FAO pour poches, contours, perçage, gravure et finition 3D en balayage parallèle	Tout le monde, gratuitement, pour apprendre, enseigner et les projets personnels qui ne rapportent pas d'argent
Maker	Évidement adaptatif, toutes les stratégies de finition 3D, reprise des restes, deux faces et plusieurs phases	Une personne, y compris pour vendre ce qu'elle fabrique
Pro	Tout Maker, plus l'imbrication, l'optimiseur de programme et le contrôle au palpeur (comparaison avec la conception) avec ses rapports	Votre organisation, pour tout usage commercial, par poste

Une licence **Propriétaire d'une MC-1** est une licence Pro pour un poste, à utiliser avec la MC-1 dont elle indique le numéro de série. Elle passe au propriétaire suivant avec la machine.

Une licence payante existe sous ces formes :

- **Abonnement** : au mois ou à l'année, jusqu'à ce que vous le résiliiez.
- **Achat définitif** : à vous pour toujours. Elle couvre toutes les versions publiées dans les 12 mois suivant l'achat, ou suivant un renouvellement payant des mises à jour.
- **Propriétaire d'une MC-1** : comme un achat définitif, lié à la machine.

Postes et ordinateurs

Un poste payant est utilisé par une seule personne à la fois, sur trois ordinateurs au plus. Un ordinateur que plusieurs personnes partagent compte une fois.

Chaque activation lie la licence à un ordinateur par une empreinte hachée de cet ordinateur. L'empreinte elle-même ne quitte jamais l'ordinateur. Pour voir les ordinateurs d'une licence, ouvrez la page des licences, twinwright.com/fr/licence, et saisissez le code d'activation.

Essayer une édition payante

Vous pouvez essayer chaque édition payante gratuitement pendant 30 jours, une fois sur chaque ordinateur.

1. Utilisez une fonction payante, par exemple l'évidement adaptatif dans le panneau FAO. Résultat : Studio indique l'édition dont la fonction a besoin.
2. Choisissez **Essayer 30 jours…**.
3. Lisez ce que l'essai ajoute et jusqu'à quand, puis choisissez **Commencer l'essai**. Résultat : Studio demande l'essai au serveur de licences de Twinwright, et les fonctions de l'édition sont disponibles jusqu'à la date affichée.

Pour un essai, Studio envoie l'édition, une empreinte hachée de cet ordinateur et la version de Studio, rien d'autre. Le serveur accorde à chaque ordinateur un essai de chaque édition : si vous réinstallez Studio pendant l'essai, le démarrer de nouveau vous redonne la même date de fin. À la fin de l'essai, Studio redevient l'édition Learning ; vos projets et programmes restent intacts.

Sur un ordinateur sans Internet, Studio affiche à la place un code de demande. Sur un ordinateur avec Internet, ouvrez la page des licences, twinwright.com/fr/licence, choisissez l'édition sous « Essayer sur un ordinateur sans Internet », collez le code de demande et téléchargez le fichier d'essai. Importez ensuite le fichier sur le premier ordinateur, comme dans Activer un ordinateur sans accès à Internet.

Activer en ligne

Il vous faut le code d'activation reçu par e-mail avec votre licence, de la forme `TW-XXXXX-XXXXX-XXXXX-XXXXX`, et une connexion à Internet.

1. Choisissez **Aide > Licence…**.
2. Choisissez **Activer en ligne…**.
3. Saisissez le code d'activation. Majuscules, minuscules, espaces et tirets n'ont pas d'importance.
4. Choisissez **Activer**. Résultat : le fichier de licence est dans le dossier des licences, et la boîte de dialogue affiche l'édition.

L'activation n'envoie que le code d'activation, une empreinte hachée de cet ordinateur et la version de Studio.

Si la licence est déjà sur autant d'ordinateurs qu'elle le permet, libérez-en d'abord un sur la page des licences (voir Transférer une licence sur un autre ordinateur).

Activer un ordinateur sans accès à Internet

Procédez ainsi pour un ordinateur de laboratoire ou d'atelier qui n'est jamais en ligne. Il vous faut le code d'activation, un second ordinateur avec Internet et un moyen de transporter un petit fichier, par exemple une clé USB.

1. Sur l'ordinateur sans Internet, choisissez **Aide > Licence…**.
2. Choisissez **Activer hors ligne…**.
3. Choisissez **Copier**. Résultat : le code de demande est dans le presse-papiers. Il contient une empreinte hachée de cet ordinateur et la version de Studio, aucune donnée personnelle.
4. Collez le code de demande dans un fichier texte et apportez-le à l'ordinateur connecté.
5. Sur celui-ci, ouvrez twinwright.com/fr/licence. Sous « Un ordinateur sans accès à Internet », collez le code d'activation et le code de demande.
6. Choisissez « Télécharger le fichier de licence ». Résultat : un fichier nommé comme `twinwright-TWL-….twlic` est téléchargé.
7. Rapportez le fichier. Dans Studio, choisissez **Aide > Licence…** puis **Importer un fichier de licence…**, et ouvrez le fichier. Résultat : le fichier de licence est dans le dossier des licences.

Un ordinateur sans Internet ne renouvelle pas sa licence tout seul. Répétez ces étapes :

- avant la fin d'une période d'abonnement ;
- pour un achat définitif, une fois après les 30 premiers jours (la période de remboursement), pour obtenir le fichier permanent ;
- après l'achat d'années de mises à jour supplémentaires.

Le dossier des licences

Le dossier des licences appartient à l'ordinateur, pas à un compte Windows. Une licence qui s'y trouve vaut pour tous ceux qui utilisent l'ordinateur.

Système	Dossier
Windows	`%ProgramData%\Twinwright\licence`
Linux	`/etc/twinwright/licence` ; là où l'on ne peut pas y écrire : `~/.local/share/Twinwright/licence`

La variable d'environnement `TWINWRIGHT_LICENCE_DIR` désigne un autre dossier. Pour ouvrir le dossier, choisissez **Aide > Licence…** puis **Ouvrir le dossier des licences**.

Le dossier contient les fichiers de licence (`*.twlic`, y compris ceux des essais) et un fichier avec la dernière heure à laquelle Studio a vérifié. Il peut contenir plusieurs fichiers de licence, par exemple une licence de propriétaire d'une MC-1 et un abonnement. L'édition en vigueur est la plus haute de tous les fichiers de licence valides.

> **Remarque**
>
> Ne modifiez pas ces fichiers. Un fichier de licence modifié ne compte plus : téléchargez-le à nouveau depuis la page des licences et importez-le.

Si Studio ne peut pas créer le dossier ou y écrire, un administrateur le crée et donne aux utilisateurs le droit d'y écrire. Sous Linux, importez en tant que root (`sudo`), ou définissez `TWINWRIGHT_LICENCE_DIR`.

Renouvellement et 14 jours de grâce

Avec **Renouveler avant l'échéance (n'envoie que le numéro de licence, une empreinte hachée de cet ordinateur et la version de Studio)** activé, Studio renouvelle la licence tout seul lorsqu'il est en ligne :

- un abonnement, à partir de cinq jours avant la fin de sa période ;
- un achat définitif, à la fin de la période de remboursement, et à partir de 14 jours avant la fin de ses mises à jour si vous avez acheté des années supplémentaires.

Studio essaie une fois au démarrage, au plus une fois par jour. Un essai qui échoue ne vous interrompt pas. Seul ce que l'option nomme est envoyé.

Si un abonnement n'est pas renouvelé, tout fonctionne encore pendant 14 jours. La licence affiche **En période de grâce** et les jours restants. Ensuite, Studio redevient l'édition Learning.

Quand une licence prend fin

L'échéance, la résiliation ou un échec de vérification ne font que ramener Studio à l'édition Learning :

- vos projets s'ouvrent comme avant ;
- le G-code déjà généré peut toujours être exporté et envoyé à la machine ;
- les opérations payantes attendent votre renouvellement.

Studio ne supprime jamais vos fichiers et ne verrouille jamais vos données.

Mises à jour d'un achat définitif

Un achat définitif couvre les versions publiées dans les 12 mois suivant l'achat, ou suivant un renouvellement payant des mises à jour. Une version plus récente fonctionne comme l'édition Learning, et la licence affiche **Mises à jour terminées**. Continuez avec la dernière version couverte, ou prolongez les mises à jour d'un an sur la page des licences. Voir Mettre à jour Studio.

Transférer une licence sur un autre ordinateur

1. Sur la page des licences, saisissez le code d'activation et choisissez « Afficher mes ordinateurs ».
2. Choisissez « Libérer cet ordinateur » à côté de l'ordinateur que vous n'utilisez plus.
3. Activez la licence sur le nouvel ordinateur.

Vous pouvez libérer jusqu'à trois ordinateurs en 30 jours. Au-delà, écrivez à hello@twinwright.com.

Un nouveau matériel ou un système réinstallé change l'empreinte. Studio indique alors **Cette licence est liée à un autre ordinateur.** Libérez l'ancienne entrée, puis activez à nouveau. En attendant, Studio fonctionne comme l'édition Learning.

Code d'activation perdu

Utilisez **Code d'activation perdu ?** sur la page des licences. Si elle demande une adresse e-mail, saisissez celle utilisée pour l'achat : chaque licence achetée avec cette adresse reçoit un nouveau code par e-mail. Si elle vous demande de nous écrire, écrivez à hello@twinwright.com depuis l'adresse à laquelle votre code d'activation a été envoyé ; nous vérifions et nous vous envoyons un nouveau code. Dans les deux cas, l'ancien code ne permet plus d'activer, et les ordinateurs déjà activés continuent de fonctionner.

L'horloge de l'ordinateur

Si l'horloge est reculée de plus de deux jours, Studio indique **L'horloge de cet ordinateur indique une heure antérieure à la dernière vérification de Studio.** C'est l'heure la plus tardive qui compte : aucune licence ni aucun essai ne regagne de jours. Réglez correctement la date et l'heure.

Studio garde la dernière heure vue à deux endroits. C'est la plus tardive des deux qui compte : supprimer l'une d'elles ne rend aucun jour.

Si une licence ou un essai apparaît terminé trop tôt après que l'horloge a été réglée une fois sur une date future, remettez la date et l'heure à jour et écrivez à hello@twinwright.com.

Pour les administrateurs : la ligne de commande

Avec ces options, le programme n'ouvre pas de fenêtre. Le dossier est facultatif ; sans lui, c'est le dossier des licences de l'ordinateur qui est utilisé.

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

- `--licence-status` liste chaque fichier de licence et chaque essai, s'il est valable et pourquoi ; `--json` donne tous les champs.
- `--licence-request` affiche le code de demande de cet ordinateur.
- `--licence-import` vérifie un fichier de licence et le copie dans le dossier des licences.

Sous Linux, le programme s'appelle `twinwright-studio`. Les clients IA peuvent lire le même état avec l'outil MCP en lecture seule `licence_status`.

Ce que Studio envoie et ce que notre serveur conserve : Confidentialité et ce que Studio envoie.

Vos données et sauvegardes

Studio garde votre travail dans vos documents et l'état des machines dans un dossier qui lui est propre. Ce chapitre indique ce qui se trouve où, ce qu'il faut sauvegarder et comment passer sur un autre ordinateur.

Ce qui se trouve où

Deux dossiers contiennent tout ce qui vous appartient :

- `Documents\Twinwright Studio` : votre travail ;
- `%LOCALAPPDATA%\Twinwright\Studio` : l'état propre du programme.

Travaux, programmes et leurs versions

`Documents\Twinwright Studio\programs\<program>\` contient un fichier pour chaque version d'un programme (`v0001.nc`) et une note indiquant quand, par qui et pourquoi elle a été enregistrée (`v0001.json`). Le travail FAO qui a produit un programme est versionné à côté (`job-v0001.json`). Studio ne les purge jamais.

Outils > Paramètres… > Fichiers permet de placer le dossier des programmes ailleurs (au prochain démarrage).

Rapports

`Documents\Twinwright Studio\Reports\` : rapports de contrôle (PDF et CSV), fiches de bridage (HTML) et archives de diagnostic (ZIP).

Les machines et leurs mondes

Chaque machine a un monde dans le dossier d'état : la table, la pièce avec ses usinages, les outils et la table d'outils, les décalages d'origine, les réglages du contrôleur, les températures, les heures de fonctionnement et l'étalonnage du palpeur. Studio l'écrit pendant que vous travaillez, chaque fois que la machine est restée immobile deux secondes après un changement, puis encore à la fermeture de la fenêtre. Le monde de la première machine se trouve dans `world\`, ceux des autres dans `machines\<id>\world\`.

Paramètres

`settings.json`, dans le dossier d'état, contient l'ensemble de **Outils > Paramètres…** ; `window.json`, la position et la taille de la fenêtre.

Le journal d'audit

`Documents\Twinwright Studio\logs\` contient un fichier par jour (`audit-YYYY-MM-DD.jsonl`). Les enregistrements sont chaînés : une modification ou une lacune apparaît lorsque vous vérifiez le journal (**Outils > Paramètres… > Journal > Vérifier l'intégrité**). Par défaut, le journal conserve 180 jours et 2 Go au plus ; **Outils > Paramètres… > Journal** permet de le changer. Copiez le dossier entier pour que la chaîne reste complète.

Ce qu'il faut sauvegarder

Fermez Studio, puis copiez les deux dossiers : `Documents\Twinwright Studio` et `%LOCALAPPDATA%\Twinwright\Studio`.

Deux choses ne suivent pas avec une copie :

- les clés des services d'IA : elles sont chiffrées pour votre compte Windows sur cet ordinateur. Saisissez-les de nouveau sur le nouvel ordinateur ;
- la licence : elle appartient à l'ordinateur. Activez-la de nouveau sur le nouveau (voir Licences).

Passer sur un autre ordinateur

1. Sur l'ancien ordinateur, fermez Studio et copiez les deux dossiers.
2. Installez Studio sur le nouvel ordinateur et lancez-le une fois, puis fermez-le.
3. Copiez les deux dossiers aux mêmes emplacements sur le nouvel ordinateur.
4. Si vous avez déplacé le dossier des programmes ou celui du journal dans les paramètres, vérifiez que ces dossiers existent.
5. Lancez Studio, activez la licence et saisissez de nouveau les clés d'IA.
6. Connectez de nouveau vos clients IA (**IA > Connecter des clients IA…**).

Repartir de zéro avec une machine

Une réinitialisation d'usine remet la machine de votre fenêtre dans l'état où elle est sortie d'usine : son monde disparaît (la pièce, les outils, les décalages d'origine, les réglages du contrôleur, l'étalonnage du palpeur) et Studio redémarre.

1. Choisissez **Fichier > Rétablir l'état d'usine…** (à partir du niveau Technicien).
2. Dans les 5 secondes, choisissez de nouveau cette commande pour confirmer.

Seule cette machine est réinitialisée. Vos programmes, vos rapports, le journal d'audit, les paramètres et les autres machines restent en place. La réinitialisation est consignée dans le journal d'audit.

Mettre à jour Studio

Studio ne cherche pas lui-même de nouvelles versions et ne se met jamais à jour tout seul. C'est vous qui décidez quand mettre à jour.

Savoir s'il existe une nouvelle version

1. Dans Studio, choisissez **Aide > À propos**. Résultat : la ligne sous le nom du programme indique votre version, par exemple « Version 1.0.0 · compilée le 2026-10-10 · commit 4f1c2ab ».
2. Ouvrez la page de téléchargement, twinwright.com/fr/download. Résultat : la page indique la version actuelle et la somme de contrôle SHA-256 du programme d'installation.

Si les versions diffèrent, la page de téléchargement propose la plus récente.

Installer une mise à jour

Il vous faut le nouveau programme d'installation. Les droits d'administrateur ne sont pas nécessaires : Studio s'installe pour votre compte Windows.

1. Téléchargez le programme d'installation depuis la page de téléchargement.
2. Facultatif : comparez sa somme de contrôle SHA-256 avec celle de la page. Dans PowerShell, exécutez `Get-FileHash` avec le nom du fichier.
3. Lancez le programme d'installation. Si Windows indique un éditeur inconnu, ne l'exécutez pas : écrivez à hello@twinwright.com.
4. Si Studio est ouvert, fermez-le lorsque le programme d'installation le demande.
5. Suivez le programme d'installation. Résultat : la nouvelle version remplace le programme dans le même dossier.

Vos paramètres, clés d'IA, versions de programme, journal d'audit et licence restent tels quels : ils ne sont pas dans le dossier du programme. Les clients IA que vous avez connectés continuent de fonctionner, car le chemin du programme ne change pas.

Pour de nombreux ordinateurs, le programme d'installation fonctionne en mode silencieux : `setup.exe /S`. Voir Installer Studio.

Les versions couvertes par votre licence

Licence	Versions avec ses fonctions payantes
Édition Learning	Toutes les versions
Abonnement	Toutes les versions, tant que l'abonnement court
Achat définitif, propriétaire d'une MC-1	Les versions publiées dans les 12 mois suivant l'achat, ou suivant un renouvellement payant des mises à jour

Une version publiée après la fin des mises à jour d'une licence fonctionne comme l'édition Learning. La licence affiche alors **Mises à jour terminées**. Avant de mettre à jour, vérifiez la date dans **Aide > Licence…**. Conservez le programme d'installation de la dernière version couverte par votre licence.

Pour continuer à mettre à jour, prolongez les mises à jour d'un an sur la page des licences. Voir Licences, postes et activation.

Ordinateurs sans accès à Internet

1. Sur un ordinateur connecté, téléchargez le programme d'installation.
2. Apportez-le à l'ordinateur sans Internet, par exemple sur une clé USB.
3. Installez-le comme ci-dessus.

Une version plus récente ne modifie pas le fichier de licence. Renouvelez comme avant : Activer un ordinateur sans accès à Internet.

Revenir à une version antérieure

1. Sauvegardez vos fichiers avant la mise à jour.
2. Désinstallez Studio : dans les Paramètres de Windows, sous « Applications ».
3. Installez le programme d'installation de la version antérieure.

> **Remarque**
>
> Une version antérieure peut ne pas ouvrir les projets ou paramètres enregistrés par une version plus récente. Gardez la sauvegarde jusqu'à en être sûr.

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Confidentialité et ce que Studio envoie

Studio fonctionne sur votre ordinateur. Studio lui-même n'envoie que ce qu'exigent le démarrage d'un essai, l'activation et le renouvellement d'une licence payante. Il n'y a ni télémétrie, ni recherche de mises à jour, ni envoi automatique des plantages. Si vous connectez l'assistant IA à un service dans le cloud, vos messages et ce que vous joignez partent vers ce service : voir L'assistant IA et les services dans le cloud. Ce chapitre indique ce qui sort, ce qui reste sur l'ordinateur et comment le supprimer. Le texte qui fait foi est la politique de confidentialité du site web, twinwright.com/fr/privacy.

Ce que Studio envoie

Quand	À qui	Quoi
Vous commencez l'essai d'une édition payante	Le serveur de licences de Twinwright, en HTTPS	L'édition, une empreinte hachée de l'ordinateur, la version de Studio
Vous activez une licence payante	Le serveur de licences de Twinwright, en HTTPS	Le code d'activation, une empreinte hachée de l'ordinateur, la version de Studio
Studio renouvelle une licence payante (voir Renouvellement)	Le serveur de licences de Twinwright, en HTTPS	Le numéro de licence, la même empreinte hachée, la version de Studio

C'est tout ce que Studio envoie lui-même à Twinwright. L'empreinte ne quitte jamais l'ordinateur telle quelle, seule sa valeur hachée. Un ordinateur activé hors ligne n'envoie rien du tout.

Trois choses ne sortent que si vous les configurez :

- **L'assistant IA, connecté à un service dans le cloud.** Il envoie vos messages et ce que vous joignez à ce service ; voir plus bas.
- **Un client IA que vous connectez par MCP**, comme Claude ou ChatGPT. Le client envoie ce qu'il lit dans le jumeau à son propre service, selon ses conditions.
- **Une page web que Studio ouvre**, comme la page des licences. Votre navigateur l'ouvre ; Studio n'y ajoute rien.

L'assistant IA et les services dans le cloud

L'assistant IA fonctionne avec le fournisseur que vous y choisissez : un service dans le cloud comme Anthropic, OpenAI ou DeepSeek, avec votre propre compte et votre clé API, ou un modèle local.

Quand vous envoyez un message à un service dans le cloud, Studio envoie à ce service, depuis votre ordinateur et avec votre clé :

- votre message, avec la conversation qui précède ;
- la machine, le bridage, le programme (avec les numéros de ligne) et les problèmes trouvés par les vérifications dans la fenêtre, tant que **Joindre le programme, le bridage et les résultats de vérification de la fenêtre (s'ils ont changé)** est coché. La case est cochée au départ. Studio les joint au premier message, puis seulement quand ils ont changé ;
- les résultats des outils que l'assistant utilise dans Studio, comme la vérification ou la simulation d'un programme, l'état de la machine ou des entrées du journal, et des images de résultats pour les services qui acceptent les images.

Avant que le premier message parte vers un service dans le cloud, Studio montre une fois ce qu'il va envoyer.

Le service traite tout cela selon ses propres conditions ; Twinwright n'en reçoit rien. Pour envoyer moins, décochez la case avant d'envoyer, ou recommencez avec **Nouvelle conversation**.

Avec un modèle local (LM Studio, Ollama), tout cela reste sur l'ordinateur.

Sur cet ordinateur, le journal d'audit garde les 180 premiers octets de chaque message (voir Le journal d'audit), et les clés API sont stockées chiffrées (voir Ce que Studio conserve sur l'ordinateur).

Ce que Studio conserve sur l'ordinateur

Quoi	Où (Windows)	Qui le voit
Paramètres, état du jumeau, clés d'IA	`%LOCALAPPDATA%\Twinwright\Studio`	Votre compte Windows
Versions de programme	`Documents\Twinwright Studio`	Votre compte Windows
Le journal d'audit	`Documents\Twinwright Studio\logs`, sauf si les paramètres indiquent un autre dossier	Votre compte Windows
Rapports de contrôle, fiches de bridage, fichiers de diagnostic	`Documents\Twinwright Studio\Reports`	Votre compte Windows
Rapports de fermeture inattendue	`%LOCALAPPDATA%\Twinwright\Studio\crash`, supprimés après 30 jours	Votre compte Windows
Fichiers de licence, essais compris	`%ProgramData%\Twinwright\licence`	Toutes les personnes de l'ordinateur

Les clés d'IA sont stockées chiffrées pour votre compte Windows (Windows DPAPI). Les autres comptes du même ordinateur ne peuvent pas les lire.

Le journal d'audit

Le journal d'audit est un fichier par jour, `audit-YYYY-MM-DD.jsonl`, dans le dossier des journaux. Il consigne :

- ce que l'IA et vous avez fait faire à la machine ;
- les 180 premiers octets de chaque message du panneau IA ;
- les vérifications de licence : le numéro de licence, l'édition et le résultat, sans le nom du titulaire.

Les clés, jetons et mots de passe y sont écrits `***`. Les paramètres fixent ce que le journal consigne et combien de temps il le conserve. Le journal reste sur l'ordinateur ; Studio ne l'envoie jamais.

Si Studio se ferme de façon inattendue

Au moment de la panne, Studio écrit un petit rapport : où l'erreur s'est produite, l'enchaînement des appels qui y a mené et la liste des modules du programme. Il n'écrit aucun contenu de la mémoire.

Au démarrage suivant, Studio y ajoute les informations de version et les 200 dernières lignes de son journal, nettoyées :

- les chemins et noms de fichiers sont remplacés par des marqueurs ;
- les noms et les adresses e-mail sont supprimés ;
- il n'y a ni G-code, ni conversation avec l'IA, ni code de licence, ni empreinte.

Studio pose ensuite la question une fois. Vous pouvez ouvrir le dossier du rapport, enregistrer le rapport dans un fichier ou fermer le message. Rien n'est envoyé. Pour nous aider à trouver la cause, envoyez vous-même le fichier à hello@twinwright.com.

Informations de diagnostic

Quand vous nous demandez de l'aide, nous pouvons vous demander des informations de diagnostic. Dans le menu **Aide**, choisissez « Collecter les informations de diagnostic ». Studio enregistre un fichier ZIP avec les éléments cochés dans le dossier des rapports :

Élément	Coché au départ
Version, système, carte graphique, langue et unités	Oui
Paramètres, sans les clés d'IA	Oui
Le profil de la machine et les réglages `$` du contrôleur	Oui
Les 7 derniers jours du journal et les rapports de fermeture inattendue, nettoyés comme ci-dessus	Oui
L'état de la licence : édition et date de fin, sans codes ni empreinte	Oui
Le programme et le bridage actuels : les données de votre pièce	Non

Ouvrez et lisez le fichier avant de l'envoyer. Rien n'est envoyé automatiquement.

Ce qui écoute sur le réseau

Les services de Studio pour les clients IA (MCP) et pour les logiciels d'envoi n'écoutent que sur cet ordinateur (`127.0.0.1`). Les autres ordinateurs ne peuvent pas les joindre.

L'option de ligne de commande `--bind any` ouvre à votre réseau les ports réseau du jumeau, comme celui des logiciels d'envoi. Le service MCP reste local.

> ⚠ **Avertissement**
>
> En mode ombre, le port des logiciels d'envoi transmet les commandes à la machine réelle. Avec `--bind any`, toute personne de votre réseau qui atteint ce port pourrait faire bouger la machine. N'utilisez `--bind any` que pour le jumeau, sur un réseau que vous maîtrisez.

Ce que notre serveur conserve

Pour chaque licence payante, le serveur de licences conserve :

- les empreintes hachées de trois ordinateurs au plus par poste, avec la date à laquelle chacun a été vu la première et la dernière fois, pour que vous puissiez en libérer un sur la page des licences ;
- le code d'activation seulement sous forme hachée : même nous ne pouvons pas le relire ;
- pendant 90 jours, un journal des essais, activations, renouvellements et modifications, avec l'empreinte hachée de l'ordinateur, la version de Studio et votre adresse IP réduite à sa partie réseau.

Pour chaque essai, il conserve l'empreinte hachée de l'ordinateur avec l'édition et les dates de l'essai, pour que chaque ordinateur ait chaque essai une seule fois.

Rien de votre travail n'atteint le serveur.

Supprimer ce que Studio conserve

1. Libérez l'ordinateur sur la page des licences, pour qu'il ne compte plus (voir Transférer une licence sur un autre ordinateur).
2. Désinstallez Studio.
3. Supprimez les dossiers du tableau ci-dessus dont vous n'avez plus besoin. Résultat : il ne reste rien de Studio sur l'ordinateur.

La désinstallation supprime le programme, pas vos fichiers : vos projets, versions de programme, journaux et rapports restent jusqu'à ce que vous les supprimiez.

Laboratoires, salles de cours et ordinateurs partagés

Ce chapitre s'adresse à la personne qui installe Studio sur des ordinateurs que plusieurs personnes utilisent : au laboratoire, à l'atelier ou en salle de cours.

Une licence pour tout l'ordinateur

Le dossier des licences appartient à l'ordinateur, pas à un compte Windows (voir Le dossier des licences). Une licence activée sur un ordinateur vaut pour toute personne qui s'y connecte.

Un poste payant est utilisé par une seule personne à la fois, sur trois ordinateurs au plus, et un ordinateur que plusieurs personnes partagent compte une fois. Donc :

- donnez à un ordinateur partagé, comme celui placé à côté de la machine, un poste à lui ;
- les personnes qui utilisent aussi les fonctions payantes sur leurs propres ordinateurs ont besoin de postes pour ceux-ci.

Activez l'ordinateur partagé une fois, avec le compte de la personne qui s'en occupe.

Ce que chaque personne conserve

Studio s'installe pour chaque compte Windows. Chaque personne a ses propres :

- paramètres et clés d'IA : les clés sont chiffrées pour ce compte, et les autres comptes ne peuvent pas les lire ;
- versions de programme, rapports et journal d'audit, dans son propre dossier `Documents\Twinwright Studio`.

L'essai d'une édition compte une fois pour l'ordinateur, pour toutes les personnes qui l'utilisent : le serveur de licences accorde à chaque ordinateur un essai de chaque édition.

Pour installer sur de nombreux comptes et ordinateurs, voir Installer Studio.

Salles de cours

L'édition Learning est gratuite sur chaque ordinateur, pour apprendre et enseigner. Rien n'est à activer : installez Studio, et tout le jumeau, l'éditeur et la FAO pour les pièces courantes fonctionnent.

Si possible, donnez à chaque élève son propre compte Windows. Ses paramètres, programmes et journaux restent alors séparés.

Pour enseigner avec les fonctions payantes, les écoles et centres de formation peuvent obtenir des postes Pro. Écrivez à hello@twinwright.com.

Ordinateurs restaurés à chaque démarrage

Certains ordinateurs de laboratoire reviennent à une image enregistrée à chaque démarrage, ce qui vide aussi le dossier des licences. Sur un tel ordinateur :

1. Activez l'ordinateur une fois hors ligne (voir Activer un ordinateur sans accès à Internet) et conservez son fichier de licence.
2. Placez le fichier dans le dossier des licences de l'image, ou importez-le à chaque démarrage :

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

Un fichier de licence est établi pour un seul ordinateur. Chaque ordinateur de la salle a besoin de son propre code de demande et de son propre fichier.

Ordinateurs sans accès à Internet

Activez-les hors ligne et renouvelez à la main : avant la fin d'une période d'abonnement, et une fois après les 30 premiers jours d'un achat définitif. Voir Activer un ordinateur sans accès à Internet.

Quand le dossier des licences n'est pas accessible en écriture

Sous Windows, tout compte peut créer `%ProgramData%\Twinwright\licence`. Le compte qui le crée peut modifier ses fichiers ; les autres comptes peuvent les lire. Cela suffit : une personne importe la licence, et tout le monde l'utilise.

Si les règles de votre organisation ne le permettent pas :

- un administrateur crée le dossier et donne aux utilisateurs le droit d'y écrire ; ou
- la variable d'environnement système `TWINWRIGHT_LICENCE_DIR` désigne un dossier que tout le monde peut lire.

Sous Linux, le dossier est `/etc/twinwright/licence` quand votre compte peut le créer ou y écrire ; sinon Studio utilise votre propre `~/.local/share/Twinwright/licence` (sous `$XDG_DATA_HOME` s'il est défini), pour vous seul. Pour partager une licence entre les utilisateurs de l'ordinateur, créez le dossier une fois avec `sudo install -d -m 1777 /etc/twinwright/licence`.

L'IA sur un ordinateur partagé

Chaque personne connecte son propre fournisseur d'IA ou client IA, avec sa propre clé. Le journal d'audit de chaque compte consigne ce que cette personne et l'IA ont fait faire à la machine. Voir Confidentialité et ce que Studio envoie.

Technical manual

Les modèles du jumeau et leurs limites

Le jumeau est la MC-1 sous forme de logiciel : le noyau propre à Twinwright y tourne, et des modèles physiques représentent les éléments de la machine qui entourent le contrôleur. Chaque chiffre que le jumeau donne sur le monde physique est une estimation tirée d'un modèle physique. Ses paramètres proviennent de la conception de la machine, des fiches techniques des composants et de valeurs publiées ; chaque MC-1 doit être mesurée à la mise en service et son jumeau réglé d'après les mesures (prévu). Chaque section ci-dessous se termine par ce sur quoi reposent ses paramètres.

Ébauche (itération 14) : le texte est en cours de rédaction ; les sections peuvent encore s'étoffer.

Le contrôleur

Le contrôleur du jumeau est le noyau propre à Twinwright, construit à partir du même code source que le micrologiciel de la machine. Ce qui est modélisé, c'est la carte qui l'entoure : ses temporisateurs et ses impulsions de pas, ses entrées avec leur antirebond, sa liaison avec l'ordinateur (le port série USB et sa poignée de main) et le temps que prend le traitement de premier plan du contrôleur.

Repose sur : le code même du noyau ; le temps de premier plan est une estimation tant qu'il n'est pas mesuré sur la carte.

Mouvement et variateurs

Chaque axe est entraîné par un variateur pas à pas en boucle fermée et une vis à billes. Le jumeau les modélise à quatre niveaux de détail ; le modèle par défaut est un modèle à trois masses :

- la boucle de position du variateur : le ressort magnétique du moteur autour de la position commandée, son couple qui diminue avec la vitesse, le décrochage quand la charge est trop forte, l'alarme du variateur quand l'erreur de poursuite devient trop grande ;
- le frottement dans les guidages, l'écrou et les roulements, et la gravité sur Z ;
- la souplesse de la transmission entre le moteur, la vis et la table, le jeu de l'écrou, le bâti sous l'axe X ;
- l'alimentation électrique des variateurs : la chaîne d'arrêt d'urgence, le contacteur, le temps de démarrage des variateurs.

Repose sur : la conception (masses, inerties, rigidité par calcul à la main et par analyse par éléments finis), les fiches techniques des variateurs et des moteurs ; le frottement et les paramètres internes du variateur sont des hypothèses tant qu'ils ne sont pas mesurés.

Modèle de géométrie et de précision

Lorsqu'il est activé, le jumeau déplace la pointe de l'outil selon les erreurs géométriques de la machine : l'erreur de pas de chaque axe et sa partie périodique, deux erreurs de rectitude et trois erreurs angulaires agissant par leurs bras d'Abbe, l'équerrage entre les axes, le jeu de l'écrou, et la dilatation thermique des vis et de la broche. Les fins de course voient les chariots, le capteur de longueur d'outil et le palpeur voient la pointe de l'outil, comme sur la machine. Le modèle est désactivé par défaut : le jumeau nominal est idéal.

Repose sur : des tableaux d'une machine assemblée typique, tirés des classes de qualité des composants. Ce n'est pas la spécification de la machine ; ces tableaux sont remplacés par la mesure ISO 230-2 de la machine à la mise en service (prévu).

Chaleur

La broche et les vis à billes s'échauffent à l'usage et refroidissent au repos ; le jumeau conserve leurs températures d'une session à l'autre et s'en sert pour déplacer la pointe de l'outil et modifier le pas des vis.

Repose sur : des modèles thermiques du premier ordre, aux paramètres supposés tant que la montée en température de la machine n'est pas mesurée.

Broche et variateur de fréquence

Le variateur de broche est modélisé comme celui de la machine : le contrôleur dialogue avec lui en RS-485 dans le protocole propre au variateur, octet par octet à la vitesse configurée, et le variateur répond comme son manuel le décrit, y compris avec ses paramètres qui décident s'il écoute. Derrière lui : le couple du moteur déduit de son glissement, son courant et les déclenchements du variateur sur surintensité, surcharge et sous-tension, le bus continu qui maintient le variateur en fonctionnement lors d'une brève baisse de tension d'alimentation, la pompe à eau.

Repose sur : les données du variateur et du moteur ; la capacité du bus, la part magnétisante du courant et le frottement sont des hypothèses tant qu'ils ne sont pas mesurés.

La coupe

Le brut est un champ de hauteurs, avec des zones plus fines là où vous le demandez. Chaque déplacement de l'outil enlève ce que la forme de l'outil balaie ; la forme de la pièce vient donc du programme lui-même.

L'effort de coupe vient des coefficients de coupe de la matière, qui varient avec l'épaisseur du copeau et la vitesse de coupe, calculés dent par dent sur la forme réelle de l'outil (à bout plat, hémisphérique, torique, en V). À partir de l'effort, le jumeau estime la charge et la puissance de la broche, la flexion de l'outil et le risque de casse de l'outil : la contrainte de flexion à la base des goujures comparée à la résistance du carbure.

Repose sur : des données de coupe publiées pour les matières (la source de chaque matière figure dans le profil de machine), un outil affûté, une géométrie d'outil typique.

Broutement

Pour chaque coupe, le jumeau calcule la profondeur à partir de laquelle elle se mettrait à brouter à la vitesse de broche utilisée, d'après les modes de vibration de l'outil et de la machine et la géométrie de la coupe, et choisit une vitesse de broche où la coupe est stable. Il estime aussi la position finale des parois d'après la vibration forcée de la coupe.

La pointe de l'outil a trois modes dans chaque direction : la structure de la machine, le porte-outil dans la pince et la broche, et la flexion propre de l'outil. La fréquence du mode de flexion découle de la forme de l'outil (ses goujures, son col, sa queue et sa longueur de sortie) : environ 14 kHz pour une fraise de 6 mm sortie de 20 mm, plus de 100 kHz pour une fraise de 0,2 mm. Les outils de moins d'environ 1 mm broutent, s'ils broutent, sur leur propre flexion, à une profondeur qui dépend à peine de la vitesse de broche ; le jumeau calcule cette profondeur dans le domaine temporel pour chaque coupe.

Repose sur : les modes du bâti issus d'une analyse par éléments finis, un mode de porte-outil supposé et la flexion de l'outil estimée à partir de sa géométrie par la théorie des poutres ; l'amortissement des trois est supposé (3 %). Aucun n'est encore mesuré : ils doivent être mesurés par un essai de frappe sur la machine construite, et jusque-là les profondeurs stables sont des estimations. L'amortissement de la coupe elle-même n'est pas modélisé : pour les petits outils, l'indice de broutement est du côté de la sécurité.

Faux-rond

Une fraise dont l'axe est décentré par rapport à celui de la broche coupe avec sa dent la plus extérieure : les alésages sortent plus grands et les bossages plus petits (constat 187). Le jumeau coupe avec cette enveloppe ; la FAO en tient compte dans les ajustements. Un outil peut porter son propre faux-rond dans sa désignation (la classe d'une pince de précision, ou une valeur mesurée ; `tir3` : son TIR en micromètres) ; sinon, celui de la machine s'applique.

Repose sur : un faux-rond supposé tant qu'il n'est pas mesuré sur la machine.

Usure des outils

En 1.0, chaque outil est un outil neuf : la forme qu'il coupe et ses efforts sont ceux d'un outil affûté, quelle que soit la durée pendant laquelle il a coupé. Le décompte de la durée de vie qu'un programme consomme pour chaque outil est prévu pour une version ultérieure. Les outils qui coupent l'aluminium sont le plus souvent changés d'abord parce que l'aluminium colle à l'arête et que la pièce présente des bavures ou un état de surface moins bon, et non parce que la face de dépouille s'est usée.

Le palpeur

Le palpeur est un outil dont la bille commute l'entrée du palpeur lorsqu'elle touche la pièce. Le jumeau calcule exactement le contact à partir de la forme du brut et des surfaces usinées, pentes comprises. Deux modèles : un palpeur idéal qui commute dès que la bille touche, et un palpeur typique avec une pré-course qui dépend de la direction, trois lobes, une dispersion et la rupture du stylet lorsqu'on le pousse trop loin.

Repose sur : des valeurs typiques d'un palpeur à déclenchement par contact ; étalonnez le palpeur sur une bague étalon avant de mesurer avec lui.

Le dégagement en Z

Pour le bridage posé sur la table, le jumeau calcule le point le plus haut à dégager et, pour l'outil dans la broche et chaque outil de la table d'outils, la hauteur de la pointe de l'outil en haut de la course en Z. Un outil qui ne peut pas dégager le bridage est signalé quand le bridage ou l'outil change (constat 188).

Repose sur : la conception de la machine et les longueurs d'outil telles qu'elles sont données.

Collisions et maintien de la pièce

Le jumeau vérifie l'outil, son porte-outil et la broche par rapport à la pièce, au montage et à la machine, et prévient avant une quasi-collision. Le maintien de la pièce (brides, étau, vé, ruban adhésif, pièce fixée) retient la pièce avec une force ; une coupe qui pousse plus fort peut la faire glisser.

Repose sur : la géométrie de la machine et des montages tirée de la conception ; les forces de maintien sont des estimations.

Ce qui n'est pas modélisé

- la texture et la rugosité de la surface, au-delà de la position des parois ;
- l'arête rapportée, les copeaux et leur évacuation ;
- la flexion des parois minces sous la coupe ;
- la chaleur de la pièce elle-même ;
- ce qui se passe après la casse d'un outil.

Les essais de réception qui exigent une machine construite (les mesures des variateurs et du son, les efforts de coupe, la surface, la chaleur, les modes du bâti, le mode ombre avec une machine réelle) figurent dans les notes de conception du jumeau (docs/06, section 7).

Le noyau propre à Twinwright

Le noyau propre à Twinwright est le micrologiciel de contrôleur CNC que Twinwright a développé lui-même. Dans Studio, il fait tourner la machine du jumeau. Le même code est compilé pour la carte contrôleur de la MC-1.

Il parle le protocole série de Grbl 1.1, de sorte que les logiciels d'envoi de G-code fonctionnent avec lui. Les réglages, les alarmes et les erreurs gardent les numéros que les logiciels d'envoi connaissent.

Ce qu'il fait

- **Mouvement :** planification avec anticipation sur 100 blocs, vitesses de passage des angles d'après `$11`, et arcs sur le cercle exact. Le mouvement s'exécute par tranches de 1 ms. Chaque axe respecte sa vitesse maximale (`$110`–`$112`) et son accélération (`$120`–`$122`).
- **Fonctions de la machine :**
  - prise d'origine, fins de course matériels et limites logicielles, palpage et capteur de longueur d'outil ;
  - le variateur de broche (variateur de fréquence) en Modbus, et le refroidissement ;
  - changements d'outil comme `$341` les définit ;
  - la porte de sécurité avec stationnement ;
  - corrections, déplacement manuel (jog) et manivelle.
- **Programmes sur le contrôleur :** les programmes de son système de fichiers s'exécutent avec des sous-programmes en mots O et des macros (voir Mots O).
- **G-code :** RS274/NGC et LinuxCNC. Cela comprend une table d'outils, la correction de rayon d'outil, le raccordement de trajectoire, des expressions et des paramètres (voir G-code et M-code).

Le protocole série, les commandes temps réel et les commandes `$` figurent dans Protocole. Chaque réglage figure dans Référence des réglages, chaque alarme dans Codes d'alarme et chaque erreur dans Codes d'erreur.

Par rapport aux autres contrôleurs Grbl

Les logiciels d'envoi écrits pour Grbl fonctionnent avec le contrôleur. Cette section liste ce que vous remarquez lorsque vous êtes habitué à un autre contrôleur Grbl.

Les cas où le contrôleur s'arrête

Dans ces cas, certains contrôleurs Grbl continuent ; le noyau propre à Twinwright s'arrête ou déclenche une alarme.

- **Le variateur de broche s'arrête de lui-même** (un déclenchement, une surcharge ou sa propre touche d'arrêt) : l'avance s'arrête en moins de 0,2 s, et le contrôleur déclenche `ALARM:14`.
- **Le variateur de broche ne répond pas :** les axes ralentissent d'abord jusqu'à l'arrêt, de sorte que la position est conservée. Ensuite, le contrôleur déclenche `ALARM:19`.
- **Changement d'outil :**
  - La broche s'arrête toujours pour un changement d'outil.
  - Si un changement d'outil se poursuit sans confirmation, la broche et le refroidissement reviennent comme ils étaient.
- **Reprise après un arrêt d'avance ou la porte :** la broche atteint sa vitesse avant que l'outil ne bouge.
- **La porte s'ouvre de nouveau pendant que la machine revient du stationnement :** la machine s'arrête et remonte de nouveau à la hauteur de stationnement. Elle attend le prochain départ cycle.
- **Le logiciel d'envoi est perdu** (USB débranché, port fermé) : le travail s'arrête. `$1030` le commande ; il est actif par défaut.
- **Contacts du palpeur et de prise d'origine :** les parasites sur les fils ne comptent pas comme un contact.
- **Le texte UTF-8 dans un programme** (chinois, « Ø », « – ») est lu comme du texte, jamais comme une commande temps réel.
- **Un arc en format R sans corde dans son plan**, par exemple `G91 G2 X0 Z1 R5`, répond `error:33`.

Davantage de G-code

Certains contrôleurs Grbl refusent ou ignorent ces éléments ; le noyau propre à Twinwright les exécute comme les normes les définissent :

- la table d'outils (`G43 Hn`, `G43.2`, `G10 L1/L10/L11`) ;
- la correction de rayon d'outil (`G41`, `G42`) ;
- le raccordement de trajectoire (`G64 P`) et l'arrêt exact (`G61.1`) ;
- les sous-programmes en mots O et les structures de contrôle, les paramètres `#` et les expressions ;
- `G52`, les coordonnées polaires (`@`, `^`) et `M99` ;
- `G85`, `G86` et `G89` (perçage et alésage), où certains contrôleurs Grbl les acceptent et ne bougent pas ;
- ce que les post-processeurs FAO écrivent souvent : une ligne `%` avec un nom, `G49 H0`, des lignes de numéro de programme Fanuc (`O1000`) et le `M25` de Centroid.

Au plus près du programme

- **Les arcs** suivent le cercle exact. Certains contrôleurs Grbl les découpent en cordes qui s'écartent du cercle jusqu'à 4 × `$12`.
- **Les arcs en temps inverse** (`G93`) prennent le temps programmé. Certains contrôleurs Grbl les exécutent plusieurs fois plus vite.
- **Les temporisations** (`G4`) durent aussi longtemps que programmé. Certains contrôleurs Grbl les arrondissent par pas de 50 ms.
- **Perçage avec débourrage :** chaque trou reçoit les passes de débourrage dont il a besoin. Quand la profondeur est un nombre entier de passes, certains contrôleurs Grbl percent une passe entière de plus depuis la hauteur de retrait.
- **Les grands arcs près des limites logicielles** ne déclenchent pas de fausses alarmes.

- `M72` rétablit la correction de longueur d'outil enregistrée par `M70` ou `M73`. Certains contrôleurs Grbl gardent une correction définie ensuite quand la valeur enregistrée était 0 avec le même mode de correction (`G43.1`) ; après `M72`, le même programme peut donc y tourner avec une autre correction de longueur d'outil.
- **Accélération :** chaque axe respecte `$120`–`$122`, y compris sur les petits arcs. Un programme comportant un très grand nombre de petits arcs peut durer jusqu'à environ 2 % de plus que sur certains contrôleurs Grbl. Un programme ordinaire dure environ 0,1 % de moins.

Détails du protocole

- **Rapports d'état :**
  - Pendant un arrêt d'avance ou porte ouverte, l'avance indiquée dans `FS:` est 0 : les axes sont à l'arrêt.
  - La distance restante est indiquée jusqu'à l'arrêt des axes.
  - Pendant une alarme critique, `Bf:` montre le planificateur vide.
  - Lorsque l'état de l'analyseur est activé dans `$10`, `[GC:]` arrive une fois par changement.
- **Vérifications des entrées :**
  - Un mauvais numéro de réglage, tel que `$45x`, répond `error:3`.
  - Un réglage entier écrit `$1=25.0` enregistre 25 ; certains contrôleurs Grbl enregistrent 250.
  - `M6` et un mouvement sur une même ligne répondent `error:40`.
  - `G65` ou `G66` avec d'autres commandes sur une même ligne répondent `error:21`.
- **Alarmes et démarrage :**
  - Il n'y a pas de pause de 100 ms après une alarme.
  - Le premier démarrage avec un stockage de réglages vide ne répond pas `error:7`.
- **Textes et réglages :**
  - Les descriptions des réglages, des alarmes et des erreurs sont propres à Twinwright.
  - Le contrôleur a ses propres réglages : `$1030` (arrêt lorsque le logiciel d'envoi est perdu), et `$1000` et suivants pour l'input shaping (désactivé par défaut).
- `$39=0` (anciens caractères temps réel désactivés) : `?`, `!` et `~` sont du texte à l'intérieur des lignes `$` et des commentaires, et des commandes partout ailleurs. Le contrôleur décide d'après les octets au fur et à mesure qu'ils arrivent. Il répond donc à `?` aussi avec la porte ouverte ou pendant un arrêt d'avance, là où certains contrôleurs Grbl ne le font parfois pas.

Non pris en charge

- `G65 P7` (messages Modbus).
- Les variables système Fanuc. `#4120` et les autres du même genre sont des paramètres ordinaires et se lisent 0.
- Le filetage (`G33`, `G76`, `G33.1`) sans codeur de broche. La MC-1 n'en a pas : ces commandes répondent donc `error:20`.
- Les cycles d'ébauche de tournage (les autres contrôleurs Grbl n'en ont pas non plus).
- Réseau, drivers Trinamic et autres marques de variateurs de broche, que certains contrôleurs Grbl ajoutent sous forme de modules d'extension.

Logiciels d'envoi

Les logiciels d'envoi qui ont été vérifiés, et leurs réglages, se trouvent dans Logiciels d'envoi de G-code. Pour comparer une machine réelle à son jumeau, voir Machine réelle et mode ombre.

G-code et M-code

Le noyau propre à Twinwright exécute le G-code ci-dessous. Les codes suivent RS274/NGC et LinuxCNC. Un code que le contrôleur ne prend pas en charge répond par `error:20`, et la ligne ne fait rien.

Pour les commandes temps réel (des octets isolés comme `?`, `!` et `~`), voir Protocole. Pour les différences entre le contrôleur et les autres contrôleurs Grbl, voir Le noyau propre à Twinwright.

G-code

Code	Signification	Remarques
`G0`, `G1`	Déplacement rapide, avance linéaire	
`G2`, `G3`	Arc dans le sens horaire, dans le sens antihoraire (I J K ou R), hélice	L'outil suit le vrai cercle. Un arc programmé avec R et sans corde dans son plan répond par `error:33`.
`G4 P`	Temporisation, P secondes	La temporisation dure aussi longtemps qu'elle est programmée.
`G5`, `G5.1`	Spline cubique, spline quadratique	Un `G5` sans I ni J continue dans la direction où le déplacement précédent s'est terminé.
`G7`, `G8`	Mode diamètre du tour, mode rayon	Mode tour uniquement (`$32=2`).
`G10 L2`, `G10 L20`	Définir un décalage d'origine	
`G10 L1`, `G10 L10`, `G10 L11`	Définir la table d'outils	32 outils. La table est conservée quand le contrôleur est éteint.
`G17`, `G18`, `G19`	Choisir le plan	
`G20`, `G21`	Unités en pouces, unités en millimètres	
`G28`, `G30`	Aller à une position mémorisée en passant par un point intermédiaire	
`G28.1`, `G30.1`	Mémoriser la position courante	
`G33`, `G33.1`, `G76`	Mouvement synchronisé avec la broche, taraudage rigide, cycle de filetage	Ils exigent un codeur de broche. La MC-1 n'en a pas : ils répondent donc par `error:20`.
`G38.2`–`G38.5`	Palpage	Le résultat est le point où le palpeur s'est déclenché, rapporté sous la forme `[PRB:]`. Définissez l'origine pièce à partir de ce point, pas de l'endroit où les axes se sont arrêtés.
`G40`	Correction de rayon d'outil désactivée	
`G41`, `G42`, `G41.1`, `G42.1`	Correction de rayon d'outil : D tiré de la table d'outils, ou `.1` avec un diamètre	Plan G17 uniquement. Mouvements d'entrée et de sortie comme dans LinuxCNC. Un mouvement qui provoquerait une surcoupe est refusé.
`G43 Hn`, `G43.2 Hn`	Correction de longueur d'outil tirée de la table d'outils, correction supplémentaire	
`G43.1`	Correction de longueur d'outil dynamique	
`G49`	Correction de longueur d'outil désactivée	
`G50`, `G51`	Mise à l'échelle désactivée, activée	
`G52`	Système de coordonnées local	Partage son décalage avec `G92`, comme dans LinuxCNC.
`G53`	Coordonnées machine pour cette ligne	
`G54`–`G59`, `G59.1`–`G59.3`	Systèmes de coordonnées pièce	`$486` peut verrouiller `G59.1`–`G59.3`.
`G61`	Trajectoire exacte	
`G61.1`	Arrêt exact	Les axes s'arrêtent à la fin de chaque mouvement programmé.
`G64 P Q`	Lissage de trajectoire	Là où cela fait gagner du temps, le coin entre deux mouvements rectilignes devient un arc à moins de P du coin. Q est accepté et n'est pas utilisé.
`G65 P1`–`P6`	Macros intégrées	Réglages, corrections d'outil, paramètres, état de la machine, choix du palpeur, attente de la broche. `P7` (messages Modbus) n'est pas pris en charge.
`G65 P100` et au-delà, `G66`, `G66.1`, `G67`	Macros de fichier, appels de macro modaux	P doit être un entier. Aucune autre commande ne peut figurer sur la même ligne (`error:21`).
`G73`, `G81`, `G82`, `G83`	Cycles de perçage, perçage avec débourrage	Chaque trou reçoit les débourrages dont il a besoin.
`G84`, `G85`, `G86`, `G89`	Cycles de taraudage et d'alésage	
`G80`	Cycle désactivé	
`G90`, `G91`	Distances absolues, distances incrémentales	
`G91.1`	Centres d'arc incrémentaux	
`G92`, `G92.1`, `G92.2`, `G92.3`	Décalage de coordonnées : définir, effacer, suspendre, rétablir	
`G93`, `G94`, `G95`	Temps inverse, unités par minute, unités par tour	Les arcs en temps inverse prennent le temps programmé.

Code	Signification	Remarques
`G96`, `G97`	Vitesse de coupe constante, vitesse de broche constante	Mode tour uniquement.
`G98`, `G99`	Retrait du cycle à la hauteur de départ, au plan R	

Ces G-codes ne sont pas pris en charge (`error:20`) :

- `G5.2`, `G5.3` (NURBS) ;
- `G17.1`–`G19.1` (plans UVW) ;
- `G70`–`G72` (cycles de tournage) ;
- `G90.1`.

M-code

Code	Signification	Remarques
`M0`, `M1`	Pause du programme, pause optionnelle	La commande temps réel `0x88` active ou désactive les pauses `M1`.
`M2`, `M30`	Fin de programme	
`M3`, `M4`, `M5`	Broche dans le sens horaire, dans le sens antihoraire, arrêtée	
`M6`	Changement d'outil	`$341` détermine ce que fait M6. M6 et un mouvement sur la même ligne répondent par `error:40`.
`M7`, `M8`, `M9`	Refroidissement : pulvérisation, arrosage, arrêt	Sur la MC-1, M8 commande le refroidissement par eau de la broche.
`M25`	Z remonte à la hauteur de la prise d'origine	Code de Centroid, écrit par certains post-processeurs FAO.
`M48`, `M49`	Corrections activées, désactivées	
`M50`–`M53`	Correction d'avance, correction de broche, avance adaptative, arrêt d'avance : activé ou désactivé	
`M56`	Stationnement activé ou désactivé	
`M60`	Pause pour changement de palette	
`M61`	Définir l'outil courant sans changement d'outil	
`M62`–`M65`	Sortie numérique activée ou désactivée, avec le mouvement suivant ou immédiatement	Sur les cartes dotées de sorties auxiliaires.
`M66`	Attendre une entrée	L'attente commence quand les mouvements qui la précèdent sont terminés.
`M70`–`M73`	Enregistrer et restaurer l'état modal	
`M98 P L`, `M99`	Appeler un sous-programme, retour	Dans les programmes exécutés depuis le système de fichiers du contrôleur. Un `M98` envoyé ligne par ligne répond par `error:39`. Un `M99` dans le programme principal le termine.

Ces M-codes ne sont pas pris en charge (`error:20`) :

- `M19` (orientation de broche) ;
- `M67`, `M68` (sorties analogiques ; la MC-1 n'en a pas) ;
- `M100`–`M199` (M-codes utilisateur).

Mots, syntaxe et paramètres

Structure d'une ligne

- Les commentaires `( … )` et `;` sont autorisés.
- `(MSG,…)` envoie le texte au logiciel d'envoi sous forme de message.
- `(PRINT,…)` envoie un message avec les valeurs des paramètres qu'il nomme.
- `(DEBUG,…)` fait de même tant que `#5599` est activé.
- Le saut de bloc `/`, les numéros de ligne `N` et les lignes `%` (avec ou sans nom) sont autorisés.
- La casse n'a pas d'importance, et les espaces à l'intérieur des mots sont ignorés.

Paramètres et expressions

- Les paramètres numérotés `#1`–`#5000` appartiennent au programme. Ils valent 0 tant que le programme ne les a pas définis.
- `#5001` et au-delà, ainsi que les paramètres système nommés, sont en lecture seule.
- Paramètres nommés `#<name>` : la lecture d'un nom qui n'a jamais été défini répond par `error:39`. `EXISTS[#<name>]` indique s'il l'a été.
- Les expressions entre `[ ]` utilisent les opérateurs et les fonctions de LinuxCNC. Les angles sont en degrés.
- Une valeur qu'une ligne définit ne peut pas être lue dans cette même ligne.

Mots O

- Les programmes exécutés depuis le système de fichiers du contrôleur peuvent utiliser des sous-programmes, `if`, `elseif`, `else`, `while`, `do` et `repeat`.
- Les sous-programmes et les boucles envoyés ligne par ligne sont rassemblés, puis exécutés une fois complets.

- Un sous-programme peut aussi être un fichier à part : `<name>.ngc` ou `<name>.macro`.

Coordonnées polaires

`@` est le rayon et `^` l'angle, dans le plan XY autour de l'origine pièce.

Erreurs dans une ligne

- Deux codes du même groupe modal sur une même ligne répondent par `error:21`.
- Deux commandes qui utilisent toutes deux les mots d'axe sur une même ligne répondent par `error:24`.
- La liste complète des codes d'erreur figure dans Codes d'erreur.

Stratégies et paramètres de FAO

Ce que fait chaque stratégie, ce que signifient ses paramètres et comment le jumeau fixe les données de coupe. Les chiffres donnés par le jumeau sont des estimations issues de ses modèles physiques.

Les paramètres ci-dessous sont ceux qu'une entité prend dans un travail (les entités de l'outil `program_generate`). Les longueurs sont en millimètres, les angles en degrés.

Évidement d'une poche : concentrique, zigzag et adaptatif

La zone d'une poche est sa forme moins ses îlots, décalée vers l'intérieur du rayon de l'outil. Elle est évidée couche par couche, chaque couche ayant au plus la profondeur de passe (`ap`). La stratégie (`strategy`) définit le trajet dans une couche :

Stratégie	Trajet	À utiliser pour
`rings` (par défaut)	Décalages de la zone, de son cœur jusqu'à la paroi, espacés de `ae`	La plupart des poches
`zigzag`	Passes droites en allers-retours selon `angle`, espacées de `ae`	Zones ouvertes et peu profondes
`adaptive`	Un trajet qui garde partout l'engagement de l'outil près de `ae`	Poches profondes, matières plus dures

Une poche avec un ajustement finit ses parois à la cote en une passe à part. Voir Ajustements.

L'évidement adaptatif coupe avec un faible engagement radial à une `ap` plus grande. Ses liaisons se déplacent à l'écart de la matière, et il revient à la stratégie concentrique là où la zone est trop étroite pour lui.

Une passe de finition enveloppe l'outil bien davantage dans un angle intérieur de paroi que le long d'une paroi droite. Là où l'enveloppement dépasse 1,2 fois celui d'une paroi droite, cette partie de la paroi est d'abord taillée par paliers (`corner_steps`, activé par défaut ; désactivé quand l'entité donne son propre `finish_ap`, sauf si elle donne aussi `corner_steps: true` ; `corner_steps: false` le désactive).

Reprise des restes

Avec `rest: true`, une poche n'évide que ce que les outils précédents ont laissé en place : les coins et les parties étroites qu'un outil plus grand ne pouvait pas atteindre. Studio détermine ce qui reste d'après les poches précédentes du même programme, ou d'après la pièce du jumeau quand celui-ci en a une.

Contours et attaches

Un contour (`side` : `outside`, `inside` ou `on`) suit un profil couche par couche, en avalant. `outside` et `inside` décalent le profil du rayon de l'outil ; `on` le suit tel quel.

Les attaches (`tabs` : nombre, largeur et hauteur) laissent de courts ponts au fond, pour qu'une pièce découpée reste en place. Une attache mesure 1 mm de haut, sauf si vous indiquez sa hauteur ; là où la coupe fait 1 mm de profondeur ou moins, sa hauteur est la moitié de la profondeur.

Rainures : par passes et trochoïdal

Une rainure (`x0`, `y0`, `x1`, `y1`, `width`) est usinée selon la stratégie :

- `layers` (par défaut) : le milieu de la rainure à pleine largeur d'outil, par couches, puis vers l'extérieur jusqu'à sa largeur de chaque côté ;
- `trochoidal` : de petites boucles le long de la rainure, chacune avancée de `step`. L'outil n'est engagé que sur un arc court, ce qui lui permet de couper plus profond en une passe. La rainure doit être plus large que l'outil.

Comment un outil entre dans la matière : hélice, rampe, plongée

Une fraise 2 tailles ne plonge pas dans la matière pleine. Chaque couche commence par :

- une hélice, là où la zone a la place d'en faire une (son diamètre et son angle figurent dans le champ `entry` de la réponse) ;
- une rampe le long du trajet, là où ce n'est pas le cas ;
- une plongée droite seulement dans un trou ou hors du brut.

La réponse dit, pour chaque opération, comment l'outil est entré en premier.

Trous

Cycles de perçage

Les trous situés aux `points`, de profondeur `depth`, sont percés avec les cycles du contrôleur. Un trou plus profond que trois fois le diamètre du foret est percé avec débourrage, à raison de 1,5 fois le diamètre à chaque débourrage, sauf si `peck` en indique un autre. Le plan de retrait du cycle est à 1 mm au-dessus de la face supérieure.

Alésage par interpolation hélicoïdale

Un trou de diamètre `d` plus grand que la fraise est usiné en suivant une hélice, puis fini à la cote au fond. Le trou peut faire au plus deux fois le diamètre de l'outil ; les trous plus grands sont usinés en poche (`shape: circle`).

Alésage à l'alésoir

Un alésoir (`reamer 6 H7`) correspondant à l'ajustement du trou entre à son avance et ressort à la même avance, sans déplacement rapide pour sortir du trou. Le trou doit être percé avant lui dans le même travail, plus petit de 0,05–0,6 mm, ou être marqué `predrilled`. La vitesse de l'alésoir est le quart de la vitesse de coupe de la fraise 2 tailles pour la matière, et son avance par tour est 2,5 fois celle du foret.

Fraisage de filets, filetages de tuyauterie coniques compris

Une fraise à fileter descend dans l'alésage, puis usine le filet de bas en haut en suivant une hélice d'un pas par tour : filet à droite dans le sens antihoraire (en avalant), filet à gauche dans le sens horaire. Le filetage est désigné (`thread` : `M8`, `M8x1`, `1/4-20 UNC`, `G1/4`) ou donné par `d` et `pitch`. Les filetages de tuyauterie coniques (`1/4 NPT`, `Rc1/4`) sont usinés autour de leur cône, chaque tour en 48 arcs.

Gorges dans un alésage

`groove` taille une gorge annulaire dans la paroi d'un alésage (pour un circlip ou un joint torique, ou un piège à lumière) avec une fraise à rainurer en T : un disque sur un col plus fin. La fraise descend au centre de l'alésage jusqu'à la gorge, s'écarte jusqu'à 0,1 mm de la paroi et fait un tour complet (en avalant) pour chaque `step` dans la paroi, puis revient au centre et remonte. Le travail donne le diamètre de la gorge `d`, celui de l'alésage `bore`, le bord inférieur de la gorge `depth` et sa hauteur `width` (par défaut l'épaisseur du disque ; une gorge plus haute est taillée en plusieurs passes de bas en haut). Elle est refusée quand l'outil dans la broche n'est pas une fraise en T, quand le disque ne descend pas dans l'alésage avec 0,2 mm de jeu, quand la gorge est plus profonde que ce que le disque dépasse de son col, et quand la gorge n'est pas plus grande que l'alésage.

Chanfreins, gravure et gravure en V

- Chanfrein : une fraise à graver en V suit l'arête du profil à la profondeur qui donne la largeur `width`. La largeur est limitée par le cône.
- Gravure : l'outil suit des lignes, ou du texte dans une police à trait unique ou à contours, à une seule profondeur `depth`.
- Gravure en V : la fraise à graver en V suit l'axe médian de la forme. Sa profondeur en chaque point donne la largeur à cet endroit, de sorte que les angles de la forme ressortent vifs. `max_depth` aplatit les zones larges à cette profondeur.

Pour graver sur une surface courbe, indiquez `surface` : le nom d'un maillage importé par `geometry_import`. Chaque trait est échantillonné tous les 0,2 mm, l'outil est posé sur le maillage et descend de `depth` sous le point où il le touche ; sur une pente, une fraise en V touche d'abord du côté amont, si bien que nulle part la coupe ne dépasse `depth`. Un trait hors du maillage est refusé.

Le perlage, le motif de cercles qui se chevauchent des horlogers, est `perlage` : une fraise deux tailles de la largeur d'un cercle descend à `depth` (0,2 mm au plus) en chaque point d'une grille à l'intérieur d'un contour, attend `dwell` secondes (0,2 par défaut) et remonte de 0,3 mm au-dessus de la face pour aller au suivant. Les rangées sont décalées d'un demi `pitch` (par défaut 0,65 fois le diamètre de l'outil), chaque cercle recouvrant une partie du précédent. `margin` garde les centres des cercles à cette distance à l'intérieur du contour.

Ébauche et finition 3D

Un maillage est ébauché par niveaux (`rough3d`), chaque niveau étant évidé de façon adaptative. Il est ensuite fini (`finish3d`) avec :

Stratégie	Trajet
`parallel`	Passes dans une seule direction à travers la pièce
`waterline`	Contours de niveau, pour les parois raides
`steep_shallow` (par défaut)	Z constant sur les zones raides, parallèle sur les zones douces
`pencil`	Le long des angles intérieurs, après les autres

`rest_from` ne finit que ce qu'un outil précédent a laissé.

Reprise d'ébauche : avec `rest_from` (la fraise plus grande, deux tailles ou torique, qui a ébauché avant, par sa désignation ou son numéro d'outil), `rough3d` ne taille que ce que cet outil a laissé, comme les espaces entre nervures qu'il n'atteignait pas ; `rest_allowance` est la surépaisseur qu'il a laissée (par défaut celle de cette passe).

Données de coupe issues du jumeau

Studio part des données de la matière pour le type d'outil : la plage de vitesses de coupe et l'avance par dent. Les valeurs que vous donnez dans `cutting` (`rpm`, `feed`, `ap`, `ae`, `finish_ap`) sont conservées telles que vous les donnez. Les autres valeurs sont déterminées avec les modèles du jumeau.

Stabilité et broutement

Le modèle de broutement du jumeau calcule les lobes de stabilité de l'outil dans son porte-outil : la profondeur de passe qui reste stable à chaque vitesse de broche. Les lobes ne se trouvent que là où le modèle de l'outil et de la machine les place ; Studio ne vise donc pas le sommet d'un lobe : pour chaque vitesse, il prend la plus faible profondeur stable dans les 10 % autour de cette vitesse, choisit une vitesse comprise entre 0,7 et 1,8 fois celle que donne la matière et coupe au plus 80 % de cette profondeur.

Puissance, effort et flexion de l'outil

- Puissance : la coupe prend au plus environ 80 % de la puissance de la broche à cette vitesse.
- Effort : l'effort de coupe reste inférieur à la moitié de ce que le montage retient latéralement.
- Flexion : une passe de finition reste assez légère pour que l'estimation de la flexion de l'outil par le jumeau reste faible.

- Angles et arcs : l'avance sur un arc est limitée par l'accélération dont dispose la machine, afin que le contrôleur ne ralentisse pas en dessous de l'avance pour laquelle la coupe a été prévue. Les passes de finition ralentissent dans les angles intérieurs d'une poche, où l'engagement de l'outil augmente.

La boucle fermée : essayer le programme dans le jumeau

Lorsque la vérification est activée, Studio exécute le programme sur un jumeau neuf. Quand l'exécution montre du broutement, une charge trop élevée ou un outil qui fléchit trop, Studio modifie le programme et réessaie :

- des couches moins profondes,
- une autre vitesse de broche, entre 0,7 et 1,8 fois celle réglée, où la coupe reste stable même si les lobes sont un peu décalés,
- des repasses sur les parois d'un ajustement.

Chaque version essayée est enregistrée. Un long programme est essayé en arrière-plan de l'appel, et son rapport arrive plus tard (`run_status`). La réponse dit ce qui a changé et pourquoi.

Plusieurs bridages et imbrication

- Bridages : Studio écrit le programme de chaque bridage, puis vérifie les bridages dans l'ordre dans le jumeau, chacun sur la pièce que les bridages précédents ont laissée. Là où cette pièce diffère du brut, le programme du bridage est réécrit pour elle.
- Imbrication : les pièces sont placées sur la plaque d'après leurs contours, séparées par une marge. Chaque pièce garde ses entités, tournées et déplacées avec elle.

Limites

- Trois axes. Pas de quatrième axe, pas d'inclinaison de l'outil.
- Les fichiers STEP ne sont pas lus ; exportez en STL ou en 3MF depuis le logiciel de CAO.
- Les données de coupe et les vérifications du jumeau sont des estimations issues de modèles physiques de la machine, de l'outil et de la matière. Un outil ou une matière qui diffère de son modèle coupe autrement.
- Éditions : l'évidement adaptatif, les stratégies de finition 3D autres que `parallel`, la reprise des restes et les bridages multiples sont proposés à partir de l'édition Maker ; l'imbrication et l'optimiseur de programme sont dans l'édition Pro.

Taille maximale d'un travail

Au-delà de ces limites, Studio n'écrit pas le programme. Il dit pourquoi, et ce qu'il faut changer.

Élément	Limite
Bridages dans un travail	8
Imbrication	16 types de pièces, 200 exemplaires de chacun
Un maillage (STL, 3MF, OBJ, glTF)	5 millions de triangles ; fichier jusqu'à 1 Go
Un fichier SVG, DXF ou texte	64 Mo ; un dessin DXF jusqu'à 2 millions de segments
Évidement adaptatif	La grille du parcours contient environ 1,5 million de cellules. Une poche de la taille de la course de la MC-1 s'évide avec une fraise 2 tailles de 3 mm ; avec une fraise de 1 mm, utilisez la stratégie concentrique ou le zigzag, ou un pas latéral plus grand.

Un travail peut avoir jusqu'à 32 000 entités ; des trous identiques forment une seule entité avec leurs points. La liste des opérations, les raisons et les réparations de la boucle fermée couvrent les 64 premières entités ; une entité suivante qui broute est signalée, pas réparée. La fiche de bridage liste les 64 premières opérations et le dit.

La vérification du programme et la fiche de bridage lisent jusqu'à 2 millions de mouvements. Un arc compte pour ses courtes cordes : environ 50 pour un cercle complet de 3 mm de rayon, environ 220 pour un de 50 mm. Au-delà, elles disent où elles se sont arrêtées. La fiche de bridage donne toujours le temps du programme entier, mais pas les temps des opérations ni les hauteurs des outils après cette ligne. L'essai dans le jumeau vérifie le programme entier.

Durée d'écriture d'un programme par Studio, mesurée sur deux cœurs d'un PC de bureau (l'essai dans le jumeau vient ensuite) :

Travail	Durée	Mémoire
20 poches avec îlots sur une plaque de 180 x 130 mm	moins de 1 s	18 Mo
Une poche de la taille de la course, adaptatif, fraise 2 tailles de 6 mm	12 s	47 Mo
La même avec une fraise 2 tailles de 3 mm	37 s	70 Mo
1 000 caractères gravés	12 s	25 Mo
200 rondelles imbriquées sur une plaque	5 s	30 Mo
Ébauche 3D d'un relief de la taille de la course (un demi-million de triangles), fraise 2 tailles de 6 mm	8 min	240 Mo
Le même relief à partir de 3 millions de triangles	20 min	1 Go
Finition 3D parallèle de ce relief, fraise hémisphérique de 2 mm, hauteur de crête de 0,01 mm (un demi-million de triangles)	4 min	270 Mo
La même à partir de 3 millions de triangles	18 min	1 Go

Pour le travail en 3D :

- Ne faites pas un maillage plus fin que ce dont la pièce a besoin. Le temps de finition augmente à peu près dans la même proportion que le nombre de triangles.
- Sur une surface légèrement ondulée, finissez avec `parallel` ou `steep_shallow`, pas avec `waterline`. `waterline` y fait beaucoup de petites boucles : sur le relief ci-dessus, l'outil se déplaçait plus dans l'air que dans la matière, et le programme avait 2,8 millions de lignes.

Ajustements, faux-rond et méthodes de contrôle

Comment Studio usine une dimension selon un ajustement, comment il tient compte du faux-rond de la fraise, et comment il mesure une pièce avec le palpeur. Les chiffres du jumeau donnés ici sont des estimations issues de ses modèles physiques ; Studio n'annonce aucune précision de positionnement pour la machine.

Ajustements (ISO 286)

Une dimension peut porter un ajustement : le diamètre d'un trou (`"d": "Ø12 H7"`), celui d'un arbre (`"fit": "g6"` sur un contour extérieur), la largeur d'une rainure, ou les dimensions X et Y d'un rectangle. Studio prend les écarts limites de l'ajustement dans les tables de l'ISO 286-2 pour la dimension, ou les écarts limites tels que vous les écrivez (`"tol": "+0.015/0"`, en millimètres).

Usiner au milieu de la zone de tolérance

La paroi est ébauchée avec une surépaisseur, puis finie en une passe à part au milieu de la zone de tolérance de l'ajustement. Les ajustements de qualité IT6 et plus fine reçoivent une repasse : le tour de finition refait à la cote, pour enlever ce que la flexion de l'outil a laissé. La boucle fermée ajoute des repasses là où l'estimation de la flexion par le jumeau le demande.

La réponse dit, pour chaque ajustement, la zone, le diamètre auquel il a été usiné, et pourquoi.

Ajustements courants

Nom	Couple	Usage
`sliding`	H7/g6	Tourne ou glisse, faible jeu
`locating`	H7/h6	L'ajustement avec jeu le plus serré : assemblé à la main
`transition`	H7/k6	Un peu de jeu ou un peu de serrage : centrage, emmanchement léger
`press`	H7/p6	Serrage : emmanché à la presse

Un couple désigne les deux pièces : le trou prend le premier élément, l'arbre le second.

Le faux-rond de la fraise

Une fraise qui présente un faux-rond usine un trou plus grand qu'elle. Le jumeau modélise le faux-rond de l'outil dans son porte-outil comme une enveloppe. Pour une dimension avec ajustement, Studio déplace de cette enveloppe la trajectoire de finition, vers l'intérieur pour un trou, vers l'extérieur pour un arbre, de sorte que la dimension estimée par le jumeau reste le milieu de la zone de tolérance. Un trou alésé à l'alésoir n'est pas modifié : il a la dimension de l'alésoir.

L'enveloppe est l'estimation du jumeau pour son modèle de l'outil et du porte-outil. Mesurez la première pièce.

Mesurer avec le palpeur

Un contact et sa compensation

Chaque contact est une séquence de déplacements de palpage (G38) :

1. un déplacement protégé jusqu'à proximité de la surface, qui s'arrête si le palpeur se déclenche en chemin ;
2. un contact rapide jusqu'à la surface ;
3. un recul jusqu'à ce que le palpeur se libère, puis un peu plus loin ;
4. un contact lent jusqu'à la surface : seul son point de déclenchement est utilisé ;
5. un retour à son point de départ.

Le point de déclenchement est le centre de la bille. Studio ajoute le rayon effectif de la bille dans la direction du contact. Le rayon effectif et les lobes du palpeur viennent de son étalonnage sur une bague étalon de diamètre connu.

Trous, bossages, parois et fonds

- Trous et bossages : quatre ou huit contacts autour de la paroi à une même profondeur. Studio ajuste un cercle aux centres de la bille par les moindres carrés, puis ajoute le rayon effectif.
- Parois : contacts perpendiculaires à la paroi, en des points le long de celle-ci.
- Fonds et faces supérieures : contacts droit vers le bas.
- Les positions sont données dans les coordonnées du programme, à partir de l'origine pièce.

Tolérances générales (ISO 2768-1)

Une dimension sans tolérance propre prend la tolérance générale de la classe demandée (f, m, c, v), d'après le tableau de l'ISO 2768-1 pour sa plage de dimensions : dimensions linéaires, ainsi que hauteurs et angles des chanfreins.

Quand la grille du jumeau est trop grossière pour une tolérance, l'estimation du jumeau pour ce contrôle est déclarée trop grossière pour être jugée ; la valeur mesurée est jugée malgré tout.

Contrôles de forme (ISO 2768-2, première étape)

La classe H, K ou L après la classe de l'ISO 2768-1 (par exemple « mK ») ajoute des contrôles de forme, jugés d'après les tableaux de l'ISO 2768-2 :

Contrôle	Contacts	Tableau
Planéité	Un fond sur une grille de 3 × 3 ; l'étendue des points par rapport à leur plan	1
Rectitude	Une longue paroi, 3 à 8 points le long de celle-ci ; l'étendue par rapport à leur droite	1
Perpendicularité	Une paroi à une seconde profondeur ; son inclinaison sur la hauteur de la paroi, par rapport au Z de la machine	2
Circularité	Un trou ou un bossage en 8 points ; l'étendue des rayons	4 (la tolérance du diamètre)

Deux contacts sur une paroi ne se distinguent que s'ils sont assez éloignés l'un de l'autre. Une paroi trop courte pour cela, par exemple sous un chanfrein de son arête, est listée avec la raison au lieu d'une valeur.

Un palpeur étalonné seulement sur son rayon effectif a ses propres lobes. Ils apparaissent dans la circularité : le rapport indique l'ampleur que le modèle de palpeur du jumeau leur attribue.

Chanfreins et rainures en V

- Un chanfrein est touché deux fois le long de sa pente, à 0,3 et 0,7 de sa largeur à partir du profil. La droite passant par les deux centres de bille est parallèle à la pente : elle donne la largeur, la profondeur et l'angle du chanfrein.
- Une rainure en V (une gravure ou une gravure en V) est touchée deux fois sur chaque flanc. Les deux droites donnent la profondeur de la rainure à sa pointe, sa largeur en haut, son angle inclus et son décalage par rapport à la ligne le long de laquelle elle a été usinée.
- Un filetage n'est pas mesuré. Ses crêtes sont l'alésage droit, et le palpeur du jumeau ne rencontre que le dessus des cellules ; la réponse le dit.

Ce que signifie l'estimation du jumeau

Le jumeau usine sa pièce simulée avec le même programme, puis y effectue les mêmes contacts avec le même modèle de palpeur. Son estimation est ce que ce calcul donne. Elle montre l'effet propre du programme sur chaque dimension, dans la mesure où le jumeau le modélise : la flexion de l'outil, son faux-rond, les efforts de coupe.

L'estimation n'est pas l'incertitude de mesure de la machine. Le rapport ne dit rien de la précision avec laquelle la machine mesure ou se positionne.

Limites

- Le palpeur mesure ce qu'il touche. Une entité plus étroite que la bille, ou une surface que la bille ne peut pas atteindre perpendiculairement, n'est pas mesurée ; la réponse dit pourquoi.
- La pièce du jumeau est une grille. Ses estimations ne sont pas plus fines que cette grille ; là où elle est trop grossière pour une tolérance, l'estimation du jumeau n'est pas jugée.
- Les contrôles de forme sont une première étape de l'ISO 2768-2 : planéité, rectitude, perpendicularité et circularité. La symétrie et le parallélisme ne sont pas encore contrôlés.

Profils de machine

Ébauche (itération 14) : le texte est en cours de rédaction ; les sections peuvent encore s'étoffer.

Ce qu'est un profil de machine

Un profil de machine est le fichier à partir duquel le jumeau construit sa machine : un fichier par modèle de machine, au format JSON. Il contient la géométrie de la machine et ses axes, les variateurs et les vis à billes, la broche et son variateur, les capteurs et leur câblage, le maintien de la pièce, les données de coupe de chaque matière, le capteur de longueur d'outil et le palpeur, les sons, et ce avec quoi le contrôleur est compilé. Les réglages du contrôleur l'accompagnent sous forme de fichier de réglages.

Le profil enregistre d'où viennent ses valeurs (sa provenance) : la conception, une fiche technique, une source publiée, une hypothèse ou une mesure.

D'où il vient

Le profil de la MC-1 est généré à partir de la conception paramétrique de la machine : les mêmes paramètres qui produisent ses plans et sa nomenclature produisent son profil, de sorte que le jumeau et la conception de la machine ne peuvent pas diverger. Le profil conserve les sommes de contrôle des fichiers de conception dont il est issu. Un schéma en vérifie la forme ; le chapitre de référence Schéma du profil de machine liste chaque clé.

Les valeurs de votre machine

Chaque MC-1 doit être mesurée à la mise en service et son jumeau réglé d'après les mesures (prévu ; la procédure et ses outils sont écrits et testés sur le jumeau) :

- les fins de course : où ils commutent, leur hystérésis et leur répétabilité ;
- chaque axe selon l'ISO 230-2 : le jeu, le tableau des erreurs de pas, les pas par millimètre ;
- les coefficients de coupe d'après des rainures d'essai ;
- l'inertie et le frottement de la broche d'après un ralentissement en roue libre ;
- le palpeur sur une bague étalon.

Les valeurs mesurées sont enregistrées avec la date, l'outil qui les a mesurées et la manière ; le profil est reconstruit avec elles, chacune marquée comme mesurée avec la valeur de la conception conservée dans son historique, et les réglages du contrôleur qu'elles modifient sont envoyés à la machine.

Plusieurs machines

Studio peut garder plusieurs machines d'un même modèle. Chacune garde son propre monde d'une session à l'autre (sa position, sa pièce, ses outils, la mémoire de son contrôleur) et sa propre disposition de fenêtre ; le profil du modèle de machine est le même pour toutes tant qu'une machine n'a pas été mesurée.

La référence

Chaque clé du profil, avec son type, son unité et la valeur de la MC-1 : Schéma du profil de machine.

Aperçu : les consignes de sécurité de cette page n'ont pas encore été relues par un locuteur natif.

Matériel de la MC-1

Brouillon (tour 14) : le texte est en cours de rédaction ; les sections peuvent encore s'étoffer.

La MC-1 telle qu'elle est conçue : les chiffres de ce chapitre sont des valeurs de conception. Chaque machine construite doit être mesurée à la mise en service, et son jumeau réglé sur ce qui a été mesuré (voir Profils de machine).

La machine

La MC-1 est une fraiseuse de bureau à trois axes pour l'aluminium, le laiton et les plastiques techniques. Son portique est fixe ; la table se déplace en Y, le chariot du portique en X, la broche en Z.

Courses X / Y / Z	200 / 150 / 80 mm
Table	300 × 220 × 35 mm, trous taraudés M6 sur une grille de 25 mm
Dimensions, largeur × profondeur × hauteur	environ 508 × 490 × 564 mm, avec moteurs, chaînes porte-câbles et pieds de mise à niveau, sans le coffret de commande
Masse	environ 57 kg, sans le coffret de commande
Bâti	plaques d'aluminium 6061-T6, boulonnées
Guidages linéaires	rails HIWIN HGR15 (6), chariots à bride HGW15CC (12)
Vis à billes	SFU1204 (12 mm, pas de 4 mm), C7, sur supports BK10 / BF10
Moteurs	moteurs pas à pas à boucle fermée NEMA 23, 1,2–1,3 N·m (3), avec accouplements à mâchoires
Broche	800 W, refroidie par eau, Ø65 × 195 mm, pince ER11
Coffret de commande	un coffret séparé à côté de la machine, sur sa droite

Axes et variateurs

Chaque axe est un moteur pas à pas à boucle fermée qui entraîne directement sa vis à billes par un accouplement à mâchoires. Les variateurs signalent un défaut sur une seule ligne : les sorties d'alarme des trois variateurs sont câblées en série sur l'entrée de défaut moteur du contrôleur, de sorte qu'un variateur en défaut, ou qui perd son alimentation, arrête la machine avec une alarme.

Les déplacements les plus rapides de la conception sont de 3 000 mm/min en X et Y et de 1 500 mm/min en Z.

Fins de course et prise d'origine

Cinq capteurs inductifs M8 marquent les extrémités de course : un à chaque extrémité de X et de Y, chacun sur sa propre entrée, et un sur le coulisseau Z pour les deux extrémités de Z. Les capteurs sont normalement fermés : un fil coupé ou une fiche mal enfoncée se lit comme un capteur déclenché, de sorte que le contrôleur lève une alarme et refuse de faire la prise d'origine au lieu d'aller heurter la butée.

La prise d'origine commence par Z, puis X et Y, chacun vers son extrémité positive (X+, Y+, Z+). Il faut faire la prise d'origine de la machine après la mise sous tension.

Z et la pesanteur

Le coulisseau Z porte la broche, environ 6,35 kg. Tant que les variateurs sont alimentés, ils maintiennent Z. Quand ils ne le sont pas (après un arrêt d'urgence, un défaut de variateur ou une coupure de courant), Z avec la broche peut descendre lentement.

> ⚠ **Avertissement**
>
> Après un arrêt d'urgence ou un défaut de variateur, refaites la prise d'origine de la machine avant de continuer. Ne mettez pas la main sous la broche tant que les variateurs sont hors tension.

La mise en service comprend un essai variateurs hors tension : si Z descend, la machine reçoit un moteur Z avec frein, ou un contrepoids.

Broche

La broche est une électrobroche de 800 W refroidie par eau, Ø65 × 195 mm avec une pince ER11, à deux pôles, jusqu'à 24 000 tr/min. Un variateur de broche Huanyang HY01D523B (variateur de fréquence, monophasé 220 V en entrée) la pilote ; le contrôleur règle sa vitesse et lit son état par RS-485, à travers un module d'isolation. Une pompe à eau refroidit la broche.

- Le relais de défaut du variateur de broche est lui aussi câblé sur la ligne de défaut moteur : quand le variateur se déclenche, le contrôleur arrête le programme avec une alarme au lieu de faire avancer un outil arrêté dans la pièce.
- Lors d'un arrêt d'urgence, le variateur de broche perd son alimentation avec les variateurs, et la broche ralentit en roue libre (elle n'est pas freinée).

Contrôleur et câblage

Le contrôleur est un Teensy 4.1 sur une carte porteuse T41U5XBB, à entrées optoisolées. Il exécute le noyau propre à Twinwright, compatible avec le protocole série Grbl 1.1, et se connecte à l'ordinateur par USB.

Entrée	Ce qui y est raccordé
Fins de course	X−, X+, Y−, Y+ et Z, cinq entrées
Palpeur	le capteur de longueur d'outil et le palpeur
Arrêt d'urgence	le second contact du bouton d'arrêt d'urgence
Défaut moteur	les sorties d'alarme des trois variateurs et le relais de défaut du variateur de broche, en série
Départ cycle, Arrêt d'avance	les boutons SB2 et SB3 du coffret de commande

Alimentation

Le secteur passe par un disjoncteur. Une alimentation 24 V est toujours active : elle alimente le contrôleur, les capteurs et le circuit de commande, de sorte que le contrôleur reste connecté à l'ordinateur et signale l'arrêt d'urgence. Un contacteur (KM1) commute l'alimentation 48 V des variateurs, du variateur de broche et de la pompe.

Les boutons du coffret de commande

Quatre boutons Ø22 sur la porte du coffret de commande, vus de face :

	gauche	droite
rangée du haut	SB1 mise sous tension (vert)	SB0 arrêt d'urgence (rouge)
rangée du bas	SB2 départ cycle (vert)	SB3 arrêt d'avance (jaune)

- **SB1 mise sous tension** ferme KM1, qui s'auto-maintient ensuite. Après un arrêt d'urgence ou une coupure de courant, la machine ne se remet pas sous tension toute seule : relâchez l'arrêt d'urgence et appuyez de nouveau sur SB1.
- **SB3 arrêt d'avance** fait ce que fait l'arrêt d'avance (`!`) de votre logiciel d'envoi : les axes s'arrêtent de façon contrôlée, la broche continue de tourner. Il est placé sous l'arrêt d'urgence : une main qui rate l'arrêt d'urgence tombe donc sur l'arrêt d'avance, pas sur le départ cycle.
- **SB2 départ cycle** reprend, comme le départ cycle (`~`) de votre logiciel d'envoi.

> **Attention**
>
> L'arrêt d'avance n'est pas une fonction de sécurité. Pour arrêter la machine pour des raisons de sécurité, utilisez l'arrêt d'urgence.

Capteur de longueur d'outil et palpeur

Le capteur de longueur d'outil est à l'arrière gauche de la table, fixé par deux vis M6 dans la grille ; il occupe le coin de la zone de travail à environ X 0–57, Y 130–150 mm. Le capteur de longueur d'outil et un palpeur partagent l'entrée palpeur du contrôleur.

Avec Z en haut, la pointe d'un outil qui dépasse de 15,5 mm de la pince se trouve à 80 mm au-dessus de la table, et le nez de la broche à 130 mm. La pointe d'un outil plus long est plus basse de sa longueur supplémentaire. Un palpeur tenu dans une pince ER mesure en général 60–90 mm, de l'écrou à la pointe du stylet : avec Z en haut, un tel palpeur ne passe que sur des bridages bas, à environ 20 mm au-dessus de la table pour un palpeur de 70 mm.

Studio le vérifie pour vous : quand vous préparez le travail, changez d'outil ou modifiez la table d'outils, il compare le point le plus haut du bridage à la pointe de chaque outil avec Z en haut, et dit quel outil ne passe pas et de combien.

Maintien de la pièce

Les trous M6 de la table, sur une grille de 25 mm, reçoivent des brides, une plaque de base ou un étau. Le jumeau connaît un étau autocentrant de 75 mm sur une plaque de base en aluminium de 10 mm, des cales parallèles, des brides, du ruban adhésif double face et un vé.

- Avec l'étau sur sa plaque de base, le haut des mors est à environ 66 mm au-dessus de la table : une pièce tenue dans l'étau monte donc à environ 71 mm de haut, ou plus.
- Pour les pièces que vous voulez palper, utilisez un bridage bas : des brides sur la plaque de base (le dessus du brut à 15–30 mm au-dessus de la table), ou un étau bas. Un palpeur ne passe pas au-dessus d'une pièce tenue dans l'étau de 75 mm.
- Retirer la plaque de base de 10 mm donne 10 mm de place en plus aux outils ; pour un palpeur, ce n'est pas assez.

Sécurité

- **Arrêt d'urgence (SB0)** : il ouvre le circuit de la bobine du contacteur, ce qui coupe l'alimentation des variateurs, du variateur de broche et de la pompe, et le signale au contrôleur par son second contact. Les axes et la broche perdent leur alimentation ; la broche ralentit en roue libre ; Z peut descendre lentement (voir Z et la pesanteur). Relâcher l'arrêt d'urgence ne redémarre rien : appuyez sur SB1, attendez le démarrage des variateurs, réinitialisez le contrôleur et faites la prise d'origine.
- **Défauts** : un défaut de variateur, un déclenchement du variateur de broche ou un fil de fin de course coupé arrête la machine avec une alarme.
- L'**Arrêt d'avance** est un arrêt contrôlé pour le travail, pas pour la sécurité.

- La MC-1 n'a pas de carter de protection. Restez à distance de la broche tant qu'elle tourne et tant que les axes peuvent bouger, portez une protection oculaire, et montez des soufflets ou des protections à brosses sur les vis X et Z pour les protéger des copeaux.
- Aucune des fonctions de sécurité de la machine n'utilise l'IA, et l'assistant IA de Studio n'est pas une fonction de sécurité.

Ce qui est mesuré à la mise en service

Les valeurs de conception ci-dessus sont à vérifier sur chaque machine construite, et son jumeau est à régler sur ce qui a été mesuré :

- le point de déclenchement de chaque fin de course (des capteurs inductifs balayés par le côté peuvent se déclencher environ 2 mm plus tard que sur le dessin), puis le dégagement de la prise d'origine et les limites logicielles ;
- les sorties d'alarme des variateurs : en débrancher une, ou couper le 48 V, doit donner une alarme ;
- l'essai de Z avec les variateurs hors tension ;
- les paramètres du variateur de broche, une montée en vitesse à vide depuis le panneau du variateur, puis la commande par RS-485 ;
- le coffret de commande : l'arrêt d'urgence et SB1 comme ci-dessus, chaque fin de course et le capteur de longueur d'outil débranchés un par un (chacun doit donner une alarme), et les boutons du pupitre.

Comment le jumeau prend en compte les valeurs mesurées : Profils de machine.

Reference

Cette référence est générée à partir du programme et n'existe qu'en anglais.

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`.

Référence des réglages

Tous les réglages du contrôleur. `$<n>` en affiche un, `$<n>=<value>` le modifie, et `$$` les liste tous. Les valeurs indiquées sont celles de la MC-1. Un réglage marqué *redémarrage* prend effet après le redémarrage du contrôleur.

Sommaire

- Général
- Signaux de commande
- Limites
- Refroidissement
- Broche
- Changement d'outil
- Prise d'origine
- Palpage
- Stationnement/Porte de sécurité
- Déplacement manuel (jog)
- Modbus
- Moteurs pas à pas
- Variateur de fréquence
- Axe X
- Axe Y
- Axe Z

Général

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$10`	Options du rapport d'état		champ de bits		15871	Ce que contient le rapport d'état temps réel. L'état de l'analyseur est envoyé sur une ligne à part, seulement quand il a changé. Bits : 0 Position en coordonnées machine, 1 État du tampon, 2 Numéros de ligne, 3 Avance et vitesse de broche, 4 État des entrées, 5 Décalage d'origine (WCO), 6 Corrections, 7 Coordonnées de palpage, 8 Synchronisation du tampon au changement de WCO, 9 État de l'analyseur, 10 Sous-état d'alarme, 11 Sous-état d'exécution, 12 Activation pendant la prise d'origine, 13 Distance restante
`$11`	Écart de jonction	mm	décimal		0.002	Distance dont la trajectoire peut s'écarter dans un angle entre deux déplacements ; elle détermine la vitesse dans l'angle. Plus la valeur est petite, plus c'est lent et précis.
`$12`	Tolérance d'arc	mm	décimal		0.002	Précision avec laquelle les arcs sont suivis.
`$13`	Rapport en pouces		booléen		0	Les positions et les vitesses des rapports sont en pouces.
`$28`	Distance de dégagement G73	mm	décimal		0.100	Le court dégagement entre les débourrages de G73.
`$32`	Mode de fonctionnement		choix		0	Fraiseuse, laser ou tour. Choix : 0 Normal, 1 Mode laser, 2 Mode tour
`$39`	Activation des anciennes commandes temps réel		booléen		1	« ? », « ! » et « ~ » agissent partout comme commandes temps réel, y compris dans les commandes $ et les commentaires.
`$60`	Rétablissement des corrections		booléen		0	Les corrections reviennent à 100 % à la fin d'un programme.
`$62`	Activation de la veille		booléen		0	Passer en veille après une longue période d'inactivité.
`$63`	Actions lors de l'arrêt d'avance		champ de bits		3	Ce que fait un arrêt d'avance en plus d'arrêter le mouvement. Bits : 0 Désactivation du laser pendant l'arrêt d'avance, 1 Rétablissement de l'état de la broche et du refroidissement à la reprise, 2 Mise au minimum de la vitesse de broche (sauf M4 laser)
`$64`	Alarme forcée à l'initialisation		booléen		0	Démarrer en état d'alarme après la mise sous tension.
`$384`	Désactivation de la conservation de G92		booléen		0	Les décalages G92 ne sont pas conservés après une mise hors tension.
`$398`	Blocs du tampon du planificateur *(redémarrage)*		entier	30–1000	100	Nombre de déplacements que le planificateur examine à l'avance.
`$481`	Intervalle des rapports automatiques *(redémarrage)*	ms	entier	100–1000	0	Envoyer des rapports d'état à cet intervalle sans qu'ils soient demandés (0 : désactivé).
`$484`	Déverrouillage requis après un arrêt d'urgence		booléen		1	Après un arrêt d'urgence, la machine reste verrouillée jusqu'au déverrouillage ($X) ou à la prise d'origine.

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$486`	Verrouillage des systèmes de coordonnées		champ de bits		0	Systèmes de coordonnées que G10 ne peut pas modifier. Bits : 0 G59.1, 1 G59.2, 2 G59.3
`$676`	Actions lors de la réinitialisation		champ de bits		15	Ce qu'efface une réinitialisation. Bits : 0 Effacement de l'état référencé si la position est perdue, 1 Effacement des décalages (sauf G92), 2 Effacement de la correction de rapide, 3 Effacement de la correction d'avance
`$1030`	Arrêt du travail à la perte de l'hôte		booléen		1	Lorsque la liaison du logiciel d'envoi qui exécute un travail est coupée pendant 0.1 s (USB débranché, port fermé), le travail s'arrête comme lors d'un arrêt (0x19) : le mouvement ralentit jusqu'à l'arrêt, la broche et le refroidissement sont coupés, les lignes envoyées qui n'ont pas encore été exécutées sont abandonnées, la position est conservée. Le logiciel d'envoi est prévenu dès qu'il est de retour. Un programme lancé depuis le système de fichiers continue de s'exécuter sans son logiciel d'envoi.

Signaux de commande

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$14`	Inversion des entrées de commande		champ de bits		0	Entrées de commande inversées. Bits : 1 Arrêt d'avance, 2 Départ cycle, 6 Arrêt d'urgence, 8 Défaut moteur
`$17`	Désactivation du pull-up des entrées de commande		champ de bits		0	Entrées de commande sans la résistance de pull-up interne. Bits : 1 Arrêt d'avance, 2 Départ cycle, 6 Arrêt d'urgence, 8 Défaut moteur

Limites

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$5`	Inversion des entrées de fin de course		masque d'axes		7	Entrées de fin de course inversées, p. ex. pour des capteurs normalement fermés.
`$18`	Désactivation du pull-up des entrées de fin de course		masque d'axes		0	Entrées de fin de course sans la résistance de pull-up interne.
`$20`	Activation des limites logicielles		booléen		1	Refuser tout mouvement au-delà de la course ($130 ...). La prise d'origine est nécessaire pour que la position soit connue.
`$21`	Activation des fins de course matériels		champ de bits		1	S'arrêter immédiatement quand un fin de course se déclenche. Mode strict : aucun mouvement tant qu'un fin de course est actionné. Bits : 0 Activation, 1 Mode strict (le bit 0 active les autres)

Refroidissement

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$15`	Inversion des sorties de refroidissement		champ de bits		0	Sorties de refroidissement actives à l'état bas. Bits : 0 Arrosage, 1 Pulvérisation
`$673`	Délai de démarrage du refroidissement	s	décimal	0.5–20	0.0	Temps accordé au refroidissement pour démarrer après M7/M8.

Broche

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$30`	Vitesse de broche maximale	tr/min	décimal		24000.000	La vitesse de broche la plus élevée.
`$31`	Vitesse de broche minimale	tr/min	décimal		6000.000	La vitesse de broche la plus basse.
`$340`	Tolérance de vitesse atteinte de la broche	%	décimal		5.0	Après M3/M4, le mouvement attend que la vitesse de broche soit à ce pourcentage près de la vitesse programmée (0 : pas d'attente).
`$394`	Délai de démarrage de la broche	s	décimal	0.5–60	15.0	Temps accordé à la broche pour atteindre sa vitesse après M3/M4 (avec $340 : l'attente maximale avant une alarme).
`$539`	Délai d'arrêt de la broche	s	décimal	0.5–60	0.0	Temps accordé à la broche pour s'arrêter après M5.

Changement d'outil

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$341`	Mode de changement d'outil		choix		3	Ce que fait M6. Choix : 0 Normal, 1 Mesure d'outil manuelle, 2 Mesure d'outil manuelle à G59.3, 3 Mesure d'outil automatique à G59.3, 4 M6 ignoré
`$342`	Distance de palpage au changement d'outil	mm	décimal		70.0	Distance parcourue pour chercher le capteur de longueur d'outil.

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$343	Avance de localisation au changement d'outil	mm/min	décimal		25.0	La vitesse lente qui localise exactement le capteur de longueur d'outil.
$344	Vitesse de recherche au changement d'outil	mm/min	décimal		200.0	La vitesse rapide de recherche du capteur de longueur d'outil.
$345	Vitesse de dégagement du palpeur au changement d'outil	mm/min	décimal		200.0	La vitesse à laquelle l'outil se dégage du capteur de longueur d'outil.
$346	Options de changement d'outil		champ de bits		0	Options du changement d'outil. Bits : 0 Retour à la position précédente après M6, 1 Changement d'outil à G30, 2 Dégagement rapide du palpeur
$485	Conservation du numéro d'outil au redémarrage		booléen		0	Le numéro de l'outil dans la broche est conservé après une mise hors tension.

Prise d'origine

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$22	Cycle de prise d'origine		champ de bits		541	Le cycle de prise d'origine et ses options. Bits : 0 Activation, 1 Activation des commandes d'axe unique, 2 Prise d'origine obligatoire au démarrage, 3 Origine machine mise à 0, 4 Deux capteurs sur une même entrée, 5 Manuel autorisé, 6 Dérogation aux verrouillages, 9 Avances par axe, 10 Exécution des lignes de démarrage seulement après la prise d'origine (le bit 0 active les autres)
$23	Inversion du sens de la prise d'origine		masque d'axes		0	Axes qui font leur prise d'origine vers leur extrémité négative.
$26	Délai antirebond du capteur d'origine	millisecondes	entier		250	Temps d'attente après le déclenchement d'un capteur, avant le déplacement de prise d'origine suivant.
$27	Distance de dégagement du capteur d'origine	mm	décimal		2.000	De combien la prise d'origine recule pour dégager le capteur.
$43	Passes de prise d'origine		entier	1–128	1	Nombre de fois où le point de déclenchement est localisé.
$44	Axes de la prise d'origine, première phase		masque d'axes		4	Axes référencés en premier.
$45	Axes de la prise d'origine, deuxième phase		masque d'axes		3	Axes référencés en deuxième.
$46	Axes de la prise d'origine, troisième phase		masque d'axes		0	Axes référencés en troisième.

Palpage

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$6	Inversion des entrées du palpeur		champ de bits		1	Inverse l'entrée du palpeur, p. ex. pour un capteur de longueur d'outil normalement fermé. Bits : 0 Palpeur
$19	Désactivation du pull-up des entrées du palpeur		champ de bits		0	L'entrée du palpeur sans la résistance de pull-up interne. Bits : 0 Palpeur
$65	Options de palpage		champ de bits		0	Façon dont le mouvement de palpage est géré. Bits : 0 Correction d'avance autorisée, 1 Application des limites logicielles

Stationnement/Porte de sécurité

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$41	Cycle de stationnement		champ de bits		0	Dégager la broche quand la porte de sécurité s'ouvre. Bits : 0 Activation, 1 Désactivation à l'initialisation, 2 Activation du contrôle de dérogation du stationnement (le bit 0 active les autres)
$42	Axe de stationnement		choix		2	L'axe qui va en position de stationnement. Choix : 0 X, 1 Y, 2 Z
$56	Distance de dégagement du stationnement	mm	décimal		5.0	Distance sur laquelle l'outil est d'abord dégagé lentement.
$57	Vitesse de dégagement du stationnement	mm/min	décimal		100.0	La vitesse de ce dégagement.

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$58	Cible de stationnement	mm	décimal	≥ -100000	-5.0	Position où va l'axe de stationnement, en coordonnées machine.
$59	Vitesse rapide de stationnement	mm/min	décimal		500.0	La vitesse du reste du déplacement de stationnement.
$61	Options de la porte de sécurité		champ de bits		0	Ce qui reste en marche tant que la porte de sécurité est ouverte. Bits : 1 Maintien de l'état du refroidissement à l'ouverture de la porte
$392	Délai de démarrage de la broche	s	décimal	0.5–60	4.0	Temps accordé à la broche pour atteindre sa vitesse quand la porte se referme.
$393	Délai de démarrage du refroidissement	s	décimal	0.5–20	1.0	Temps accordé au refroidissement pour démarrer quand la porte se referme.

Déplacement manuel (jog)

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$40	Limitation des déplacements manuels		booléen		0	Les déplacements manuels s'arrêtent aux limites logicielles au lieu d'être refusés.

Modbus

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$374	Débit en bauds Modbus		choix		3	Débit en bauds de la ligne Modbus (celui du variateur de fréquence doit correspondre). Choix : 0 2400, 1 4800, 2 9600, 3 19200, 4 38400, 5 115200
$375	Délai d'attente de réception Modbus	millisecondes	entier	50–250	50	Durée d'attente d'une réponse Modbus.
$681	Format série Modbus		choix		0	Parité de la ligne Modbus (celle du variateur de fréquence doit correspondre). Choix : 0 8 bits, sans parité, 1 8 bits, parité paire, 2 8 bits, parité impaire

Moteurs pas à pas

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$0	Durée de l'impulsion de pas	microsecondes	décimal	≥ 2.0	5.0	Durée des impulsions de pas, en microsecondes. Les variateurs imposent un minimum (voir leur manuel ; ceux de la MC-1, à boucle fermée, nécessitent 2.5).
$1	Délai d'inactivité des pas	millisecondes	entier	≤ 65535	255	Durée pendant laquelle les variateurs restent activés après l'arrêt du mouvement, pour que les axes se stabilisent avant la coupure du courant. 255 les garde activés en permanence.
$2	Inversion des impulsions de pas		masque d'axes		0	Sorties de pas actives à l'état bas.
$3	Inversion du sens des pas		masque d'axes		5	Axes dont la sortie de direction est inversée (sinon l'axe se déplace dans le mauvais sens).
$4	Inversion des sorties d'activation des moteurs pas à pas		masque d'axes		7	Sorties d'activation des variateurs actives à l'état bas (la plupart des variateurs). Avec un seul signal d'activation commun, seul X compte.
$29	Délai d'impulsion	microsecondes	décimal	≤ 20	5.0	Temps entre le positionnement des sorties de direction et l'impulsion de pas.
$37	Moteurs pas à pas à maintenir activés		masque d'axes		7	Variateurs qui restent activés quand la machine est au repos ou en alarme.
$680	Délai d'activation des moteurs pas à pas	ms	entier	≤ 500	0	Temps entre l'activation des variateurs et le premier pas.

Variateur de fréquence

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$460	Adresse Modbus du variateur de broche		entier	≤ 255	1	L'adresse Modbus du variateur de fréquence.

Axe X

Réglage	Nom	Unité	Type	Plage	MC-1	Description
$100	Résolution de déplacement de l'axe X	pas/mm	décimal		800.00000	Pas par millimètre de l'axe : pas par tour du moteur multipliés par les micropas, divisés par le pas de la vis.

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$110`	Vitesse maximale de l'axe X	mm/min	décimal		3000.000	La vitesse la plus élevée de l'axe, y compris en rapide.
`$120`	Accélération de l'axe X	mm/s²	décimal		300.000	L'accélération de l'axe.
`$130`	Course maximale de l'axe X	mm	décimal		202.000	La course de l'axe à partir de sa position d'origine, pour les limites logicielles.
`$180`	Avance de localisation de la prise d'origine de l'axe X	mm/min	décimal		50.0	La vitesse lente qui localise exactement le point de déclenchement du capteur de l'axe.
`$190`	Vitesse de recherche de la prise d'origine de l'axe X	mm/min	décimal		1200.0	La vitesse rapide de recherche du capteur de l'axe.
`$1000`	Input shaper de l'axe X		choix		0	Met en forme le mouvement de l'axe pour qu'il n'excite pas une résonance de la machine à la fréquence ci-dessous : ZV est exact à cette fréquence, les autres suppriment une bande plus large, avec moins d'exactitude. Le mouvement se termine avec un retard pouvant atteindre 1.5 période. Choix : 0 Aucun, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI
`$1010`	Fréquence de l'input shaper de l'axe X	Hz	décimal	15–500	100.0	La résonance que l'input shaper de l'axe élimine.
`$1020`	Taux d'amortissement de l'input shaper de l'axe X		décimal	≤ 0.5	0.100	Le taux d'amortissement de cette résonance.

Axe Y

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$101`	Résolution de déplacement de l'axe Y	pas/mm	décimal		800.00000	Pas par millimètre de l'axe : pas par tour du moteur multipliés par les micropas, divisés par le pas de la vis.
`$111`	Vitesse maximale de l'axe Y	mm/min	décimal		3000.000	La vitesse la plus élevée de l'axe, y compris en rapide.
`$121`	Accélération de l'axe Y	mm/s²	décimal		300.000	L'accélération de l'axe.
`$131`	Course maximale de l'axe Y	mm	décimal		152.000	La course de l'axe à partir de sa position d'origine, pour les limites logicielles.
`$181`	Avance de localisation de la prise d'origine de l'axe Y	mm/min	décimal		50.0	La vitesse lente qui localise exactement le point de déclenchement du capteur de l'axe.
`$191`	Vitesse de recherche de la prise d'origine de l'axe Y	mm/min	décimal		1200.0	La vitesse rapide de recherche du capteur de l'axe.
`$1001`	Input shaper de l'axe Y		choix		0	Met en forme le mouvement de l'axe pour qu'il n'excite pas une résonance de la machine à la fréquence ci-dessous : ZV est exact à cette fréquence, les autres suppriment une bande plus large, avec moins d'exactitude. Le mouvement se termine avec un retard pouvant atteindre 1.5 période. Choix : 0 Aucun, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI
`$1011`	Fréquence de l'input shaper de l'axe Y	Hz	décimal	15–500	100.0	La résonance que l'input shaper de l'axe élimine.
`$1021`	Taux d'amortissement de l'input shaper de l'axe Y		décimal	≤ 0.5	0.100	Le taux d'amortissement de cette résonance.

Axe Z

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$102`	Résolution de déplacement de l'axe Z	pas/mm	décimal		800.00000	Pas par millimètre de l'axe : pas par tour du moteur multipliés par les micropas, divisés par le pas de la vis.
`$112`	Vitesse maximale de l'axe Z	mm/min	décimal		1500.000	La vitesse la plus élevée de l'axe, y compris en rapide.
`$122`	Accélération de l'axe Z	mm/s²	décimal		200.000	L'accélération de l'axe.
`$132`	Course maximale de l'axe Z	mm	décimal		82.000	La course de l'axe à partir de sa position d'origine, pour les limites logicielles.
`$182`	Avance de localisation de la prise d'origine de l'axe Z	mm/min	décimal		50.0	La vitesse lente qui localise exactement le point de déclenchement du capteur de l'axe.
`$192`	Vitesse de recherche de la prise d'origine de l'axe Z	mm/min	décimal		600.0	La vitesse rapide de recherche du capteur de l'axe.

Réglage	Nom	Unité	Type	Plage	MC-1	Description
`$1002`	Input shaper de l'axe Z		choix		0	Met en forme le mouvement de l'axe pour qu'il n'excite pas une résonance de la machine à la fréquence ci-dessous : ZV est exact à cette fréquence, les autres suppriment une bande plus large, avec moins d'exactitude. Le mouvement se termine avec un retard pouvant atteindre 1.5 période. Choix : 0 Aucun, 1 ZV, 2 ZVD, 3 MZV, 4 EI, 5 2HUMP_EI
`$1012`	Fréquence de l'input shaper de l'axe Z	Hz	décimal	15–500	100.0	La résonance que l'input shaper de l'axe élimine.
`$1022`	Taux d'amortissement de l'input shaper de l'axe Z		décimal	≤ 0.5	0.100	Le taux d'amortissement de cette résonance.

Codes d’alarme

Les alarmes du contrôleur. Une alarme arrête la machine et la verrouille jusqu’à ce que vous l’effaciez : `$X` ou la prise d’origine, selon l’alarme. `$EA` les liste sur le contrôleur lui-même.

Code	Signification
`ALARM:1`	Fin de course matériel : un fin de course a été atteint pendant le mouvement. La position est probablement perdue ; refaites la prise d’origine
`ALARM:2`	Limite logicielle : la cible du mouvement dépasse la course de la machine. La position est conservée ; vous pouvez déverrouiller
`ALARM:3`	Réinitialisation ou arrêt d’urgence pendant un mouvement. La position est probablement perdue ; refaites la prise d’origine
`ALARM:4`	Palpeur : le palpeur n’était pas dans l’état attendu au démarrage du cycle de palpage
`ALARM:5`	Palpeur : aucun contact dans la course programmée
`ALARM:6`	Prise d’origine : le cycle de prise d’origine a été réinitialisé
`ALARM:7`	Prise d’origine : la porte de sécurité a été ouverte pendant la prise d’origine
`ALARM:8`	Prise d’origine : le dégagement n’a pas libéré le fin de course. Vérifiez la distance de dégagement ($27) et le câblage
`ALARM:9`	Prise d’origine : aucun fin de course trouvé dans la distance de recherche. Vérifiez les réglages de course et le câblage
`ALARM:10`	Arrêt d’urgence. Relâchez-le, puis réinitialisez
`ALARM:11`	Prise d’origine requise : exécutez le cycle de prise d’origine ($H)
`ALARM:12`	Un fin de course est actionné. Libérez-le avant de continuer
`ALARM:13`	Protection du palpeur : le palpeur s’est déclenché en dehors d’un cycle de palpage
`ALARM:14`	La broche n’a pas atteint sa vitesse à temps
`ALARM:15`	Prise d’origine : le second fin de course d’un axe à équerrage automatique est introuvable
`ALARM:16`	L’autotest à la mise sous tension a échoué
`ALARM:17`	Un variateur de moteur signale un défaut
`ALARM:18`	Prise d’origine : les réglages de prise d’origine ne sont pas valides
`ALARM:19`	Modbus : aucune réponse ou réponse erronée d’un appareil (p. ex. le variateur de fréquence)
`ALARM:20`	L’extension d’E/S ne répond pas
`ALARM:21`	La mémoire non volatile (mémoire des réglages) est défaillante
`ALARM:22`	Un tampon a débordé

Codes d’erreur

La réponse `error:N` à une ligne que le contrôleur a refusée. La ligne n’a eu aucun effet. `$EE` les liste sur le contrôleur lui-même.

Code	Signification
`error:1`	Un mot G-code n’a pas de lettre : chaque mot est une lettre suivie d’un nombre
`error:2`	Un nombre manque ou n’est pas un nombre valide
`error:3`	Commande « $ » inconnue ou non prise en charge
`error:4`	Une valeur négative a été indiquée là où seules les valeurs positives sont autorisées
`error:5`	La prise d’origine n’est pas activée dans les réglages ($22)
`error:6`	L’impulsion de pas doit durer au moins 2 microsecondes
`error:7`	Les réglages enregistrés n’ont pas pu être lus ; les réglages concernés ont été remis à leurs valeurs par défaut
`error:8`	Cette commande « $ » n’est acceptée que lorsque la machine est au repos
`error:9`	Le G-code n’est pas accepté pendant une alarme ou un déplacement manuel
`error:10`	Les limites logicielles nécessitent la prise d’origine : activez d’abord la prise d’origine ($22)
`error:11`	La ligne dépassait la longueur acceptée par le contrôleur et n’a pas été exécutée
`error:12`	Cette valeur donnerait une fréquence de pas supérieure à celle que le contrôleur peut produire
`error:13`	La porte de sécurité est ouverte
`error:14`	Le texte est trop long pour être enregistré (ligne de démarrage ou informations de compilation)
`error:15`	La cible du déplacement manuel dépasse la course de la machine ; le déplacement manuel n’a pas été effectué
`error:16`	Une commande de déplacement manuel nécessite « = » et ne peut contenir que des mots de déplacement manuel
`error:17`	Le mode laser nécessite une sortie de broche PWM
`error:18`	Le contrôleur a été réinitialisé
`error:19`	La valeur doit être supérieure à zéro
`error:20`	Le bloc contient une commande G-code non prise en charge ou invalide
`error:21`	Deux commandes du même groupe modal dans un même bloc
`error:22`	Aucune avance n’a été définie
`error:23`	Cette commande nécessite un nombre entier
`error:24`	Deux commandes du bloc ont toutes deux besoin des mots d’axe
`error:25`	Un mot apparaît deux fois dans le bloc
`error:26`	La commande nécessite des mots d’axe et le bloc n’en contient aucun
`error:27`	Numéro de ligne invalide
`error:28`	Il manque un mot de valeur dont la commande a besoin
`error:29`	G59.1, G59.2 et G59.3 ne sont pas pris en charge
`error:30`	G53 ne fonctionne qu’avec G0 et G1
`error:31`	Le bloc contient des mots d’axe qu’aucune commande n’utilise
`error:32`	Un arc (G2/G3) nécessite au moins un mot d’axe dans son plan
`error:33`	La cible du mouvement est invalide
`error:34`	Le rayon de l’arc est invalide
`error:35`	Un arc (G2/G3) nécessite au moins un mot de décalage de centre (I, J ou K) dans son plan
`error:36`	Le bloc contient des mots de valeur qu’aucune commande n’utilise
`error:37`	G43.1 nécessite l’axe de longueur d’outil configuré
`error:38`	Le numéro d’outil est supérieur au maximum pris en charge, ou cet outil n’existe pas
`error:39`	Une valeur est hors plage
`error:40`	Interdit pendant l’attente d’un changement d’outil
`error:41`	La broche doit tourner pour ce mouvement (vitesse de coupe constante ou synchronisation avec la broche)
`error:42`	Le filetage nécessite le plan ZX (G18)
`error:43`	L’avance est supérieure au maximum
`error:44`	La vitesse de broche est hors plage
`error:45`	Un fin de course est actionné : seule la prise d’origine est autorisée

Code	Signification
`error:46`	Faites d'abord la prise d'origine de la machine ($H)
`error:47`	L'outil actuel n'est pas connu : définissez-le avec M61
`error:48`	Mots de valeur contradictoires
`error:49`	L'autotest à la mise sous tension a échoué : mettez le contrôleur hors tension, puis de nouveau sous tension
`error:50`	L'arrêt d'urgence est actif
`error:51`	Un variateur de moteur signale un défaut
`error:52`	La valeur du réglage est hors plage
`error:53`	Ce réglage n'existe pas sur ce contrôleur
`error:54`	La position de dégagement est en dessous de la profondeur de perçage
`error:55`	Deux axes à équerrage automatique ne peuvent pas faire leur prise d'origine en même temps
`error:56`	Ce système de coordonnées est verrouillé ($486)
`error:57`	Marque de début ou de fin de programme inattendue (%)
`error:58`	Le port n'est pas disponible
`error:60`	La carte SD n'a pas pu être montée
`error:62`	Le contenu du répertoire n'a pas pu être listé
`error:63`	Répertoire introuvable
`error:64`	La carte SD n'est pas montée
`error:65`	Aucun système de fichiers monté (pas de carte ?)
`error:66`	Le système de fichiers est en lecture seule
`error:71`	Expression : opérateur ou fonction inconnus
`error:72`	Expression : division par zéro
`error:73`	Expression : argument hors plage
`error:74`	Expression : argument invalide
`error:75`	Expression : erreur de syntaxe
`error:76`	Expression : le résultat n'est pas un nombre
`error:77`	Authentification requise
`error:78`	Accès refusé
`error:79`	Interdit tant qu'un événement critique est actif
`error:80`	Cette instruction de code O nécessite un programme que le contrôleur peut relire (depuis son système de fichiers)
`error:81`	Instruction de code O inconnue ou mal placée
`error:82`	Trop de blocs ou d'appels de code O ouverts en même temps
`error:83`	Plus de place pour l'instruction de code O
`error:84`	Le fichier n'a pas pu être ouvert
`error:85`	Le système de fichiers n'a pas pu être formaté
`error:86`	Le port ne peut pas être utilisé
`error:87`	Il y a un outil dans la broche
`error:88`	Il n'y a pas d'outil dans la broche
`error:89`	Le fichier n'a pas pu être supprimé
`error:253`	Erreur définie par l'utilisateur

Cette référence est générée à partir du programme et n'existe qu'en anglais.

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)

Cette référence est générée à partir du programme et n'existe qu'en anglais.

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	c
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	c
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	c
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	c
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	o
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	o
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	o
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	o
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	o
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	n
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	o
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	o
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.

Protocole

Le noyau propre à Twinwright (le contrôleur) parle le protocole série Grbl 1.1, avec les extensions qu'utilisent les logiciels d'envoi Grbl actuels. Un logiciel d'envoi écrit pour Grbl fonctionne avec lui. Les logiciels d'envoi qui ont été vérifiés, et leurs réglages, sont décrits dans Logiciels d'envoi de G-code.

Connexion

Où	Comment
Le jumeau dans Studio	Port TCP 2323 sur cet ordinateur, un seul logiciel d'envoi à la fois.
La MC-1	Son port USB natif, un port série (COM*n* sous Windows). Le débit en bauds est sans importance ; utilisez 115 200 là où un logiciel d'envoi en demande un. Le contrôleur répond dès que le logiciel d'envoi lève DTR.

Lignes et réponses

- **Fins de ligne :** une ligne se termine par LF ou CR. Une paire CR LF ou LF CR compte pour une seule fin ; deux LF font deux lignes.
- **Réponses :** le contrôleur répond à chaque ligne par `ok` ou `error:N`, dans l'ordre d'arrivée des lignes. Codes d'erreur donne la liste de tous les N.
- **Longueur de ligne :** jusqu'à 256 caractères. Une ligne plus longue reçoit la réponse `error:11`.
- **Comptage de caractères :** le tampon de réception contient 1 024 octets. Un logiciel d'envoi peut avoir autant d'octets en transit sans réponse. `Bf:` dans le rapport d'état donne l'espace libre.
- **Planificateur :** 100 blocs. `Bf:` donne aussi le nombre de blocs libres.
- **Lignes en attente :** pendant un arrêt d'avance ou avec la porte ouverte, les lignes attendent dans le tampon. Elles reçoivent leur réponse quand le cycle redémarre. Une réinitialisation, `0x19` (arrêt), `0x03` ou `0x85` (annulation du déplacement manuel) abandonne les lignes encore en attente.
- **Après une erreur :** la ligne de G-code suivante ne s'exécute pas et reçoit la même erreur. Cela dure jusqu'à une commande `$`, une ligne vide ou une réinitialisation.

Commandes temps réel

Une commande temps réel est un seul octet. Le contrôleur agit dessus dès son arrivée, même au milieu d'une ligne. Elle ne fait partie d'aucune ligne et ne reçoit pas de `ok`.

Octet	Commande	Effet
`0x18` (Ctrl-X)	Réinitialisation	Arrête tout aussitôt et vide ce qui est en file d'attente. Une réinitialisation pendant que les axes se déplacent déclenche `ALARM:3` : refaites la prise d'origine. Non prise en compte tant que l'arrêt d'urgence est actif.
`0x03` (Ctrl-C)	Réinitialisation, arrêt d'urgence actif	Comme Ctrl-X, prise en compte aussi quand l'arrêt d'urgence est actif.
`0x19` (Ctrl-Y)	Arrêt	Les axes ralentissent jusqu'à l'arrêt et les lignes encore en file d'attente sont abandonnées. La position est conservée et aucune alarme n'est déclenchée.
`ESC`, puis `0x14`	Redémarrage	Redémarre le contrôleur, sur les cartes capables de redémarrer d'elles-mêmes.
`?`	Rapport d'état	Un rapport d'état.
`0x80` (ou `0x05`)	Rapport d'état	Un rapport d'état, sauf quand les rapports automatiques sont activés.
`0x87`	Rapport d'état complet	Un rapport d'état avec tous les champs.
`~`, `0x81`	Départ cycle	Reprend après un arrêt d'avance, une ouverture de porte, `M0`/`M1` ou un changement d'outil.
`!`, `0x82`	Arrêt d'avance	Les axes ralentissent jusqu'à l'arrêt et restent à l'arrêt. Pendant un déplacement manuel (jog), il l'annule.
`0x83`	État de l'analyseur	Renvoie l'état modal, comme le fait `$G`.
`0x84`	Porte de sécurité	Agit comme l'ouverture de la porte : arrêt d'avance et, si le stationnement est configuré, stationnement.
`0x85`	Annulation du déplacement manuel	Met fin à un déplacement manuel et abandonne les lignes de déplacement manuel encore en file d'attente.
`0x88`	Arrêt optionnel	Active ou désactive les pauses `M1`.
`0x89`	Bloc à bloc	Active ou désactive le bloc à bloc.
`0x8B`	Mode manivelle	Avec une manivelle connectée : change son mode. Sinon, ignoré.
`0x8C`	Rapports automatiques	Active ou désactive les rapports d'état automatiques, lorsque `$481` en fixe l'intervalle. Sinon, ignoré.
`0x90`	Correction d'avance 100 %	
`0x91`, `0x92`	Correction d'avance +10 %, −10 %	Plage de 10 à 200 %.
`0x93`, `0x94`	Correction d'avance +1 %, −1 %	Plage de 10 à 200 %.
`0x95`, `0x96`, `0x97`, `0x98`	Correction de rapide	100 %, 50 %, 25 %, 5 %.

Octet	Commande	Effet
`0x99`	Correction de broche 100 %	
`0x9A`, `0x9B`	Correction de broche +10 %, −10 %	Plage de 10 à 200 %.
`0x9C`, `0x9D`	Correction de broche +1 %, −1 %	Plage de 10 à 200 %.
`0x9E`	Arrêt broche	Pendant un arrêt d'avance seulement : arrête la broche, ou la relance.
`0x9F`	Arrêt d'urgence logiciel	Arrête aussitôt et déclenche l'alarme d'arrêt d'urgence.
`0xA0`	Refroidissement, arrosage	Active ou désactive l'arrosage.
`0xA1`	Refroidissement, pulvérisation	Active ou désactive la pulvérisation.
`0xA3`	Changement d'outil terminé	Confirme un changement d'outil manuel.

Le contrôleur écarte les autres octets de contrôle, ainsi que ceux de `0x80` à `0xBF` qui n'ont pas de commande.

- Le **texte UTF-8** dans les commentaires et les messages est lu comme du texte. Un octet de `0x80` à `0xBF` qui appartient à un tel caractère n'est jamais pris pour une commande temps réel.
- `$39=0` (anciens caractères temps réel désactivés) : `?`, `!` et `~` sont du texte dans les lignes `$` et les commentaires, et des commandes partout ailleurs. Avec `$39=1` (la valeur par défaut), ce sont toujours des commandes.

Accueil et identité

Après la mise sous tension et après chaque réinitialisation, le contrôleur annonce :

```
Grbl 1.1f ['$' or '$HELP' for help]
```

Les logiciels d'envoi prennent une ligne qui commence par « Grbl » pour un redémarrage.

`$I` répond par ces lignes, puis `ok` :

Ligne	Ce qu'elle indique
`[VER:Twinwright 1.1f.<date>:<text>]`	La version du protocole et sa date de compilation. `<text>` est ce qu'a enregistré `$I=<text>`.
`[OPT:…]`	Les lettres d'options, les blocs du planificateur (100) et le tampon de réception (1 024).
`[NEWOPT:…]`	Les noms d'options : `ENUMS`, `RT+` ou `RT-` (`$39`), `HOME`, `ES`, `MPG`, `LATHE`, `REBOOT`, `TC`, `SS` (synchronisation de broche), `SED`, `RTC`, `SD`, `FS`, `YM`.
`[FIRMWARE:Twinwright]`	Le nom du micrologiciel.
`[SIGNALS:…]`	Les entrées de commande que la carte possède, sous forme de lettres.
`[NVS STORAGE:…]`	L'endroit où les réglages sont conservés.
`[DRIVER:…]`, `[DRIVER VERSION:…]`, `[DRIVER OPTIONS:…]`, `[BOARD:…]`	La carte.
`[KERNEL:<version>+<build>]`	La version et la compilation du contrôleur : le commit d'une version publiée, ou `dev`.
`[KNCAP:…]`	Ce que comporte la carte (ci-dessous).
`[PLUGIN:…]`	Des fonctions nommées comme les connaissent les logiciels d'envoi : `MODBUS` et `HUANYANG VFD` pour le variateur de broche.

Ce que comporte la carte : KNCAP

`[KNCAP:]` est une ligne de paires `key=value`, séparées par des virgules. Des clés ne sont ajoutées qu'à la fin, et aucune ne change jamais de sens. Lisez la ligne par clé, non par position. Une clé absente signifie « non connu ».

Clé	Valeurs	Signification
`port`	`teensy41`, `stm32h7`, `stm32f4`, `twin`, `host`, `sim`, `unknown`	De quelle compilation du contrôleur il s'agit.
`pv`	un nombre	La version de l'interface de portage du contrôleur.
`view`	un nombre	La version de la vue d'état du contrôleur.
`spindle`	`vfd`, `pwm`, `none`	Le variateur de broche : Modbus, une sortie PWM, ou aucun.
`encoder`	`0`, `1`	Un codeur de broche est installé.
`sd`	`0`, `1`	La carte a un emplacement pour carte SD.

Clé	Valeurs	Signification
`mpg`	`0`, `2`	Un port de manivelle, avec son numéro de mode.
`aux`	`di/do/ai/ao`	Entrées et sorties auxiliaires : entrées numériques, sorties numériques, entrées analogiques, sorties analogiques.
`axes`	des lettres	Les axes, par exemple `XYZ`.
`probe`	`0`, `1`	Une entrée de palpeur. Sur la MC-1, le capteur de longueur d'outil l'utilise aussi.
`link`	`usb`, `uart`, `tcp`, `none`	La liaison du logiciel d'envoi. `usb` est un port série USB qui répond dès que DTR est levé.

Le mode ombre compare `[KERNEL:]` et `[KNCAP:]` entre une machine réelle et son jumeau (voir Machine réelle et mode ombre).

Rapports d'état

Un rapport d'état arrive après `?`, `0x80` ou `0x87`. Quand les rapports automatiques sont activés (`$481`, `0x8C`), il arrive aussi de lui-même. C'est une seule ligne :

```
<Idle|MPos:0.000,0.000,0.000|Bf:100,1023|FS:0,0|Pn:XYZ>
```

L'état vient en premier :

- `Idle`, `Run`, `Jog`, `Home`, `Check` (mode vérification), `Sleep`, `Tool` (un changement d'outil) ;
- `Hold:0` (suspendu) et `Hold:1` (en train de ralentir) ;
- `Door:0` (fermée, prête à reprendre), `Door:1` (arrêtée, encore ouverte), `Door:2` (ouverte : arrêt d'avance ou stationnement) et `Door:3` (fermée, rétablissement) ;
- `Alarm`.

Les champs qui suivent dépendent de `$10` et de la situation :

Champ	Signification
`MPos:` ou `WPos:`	Position machine ou position pièce (`$10` bit 0).
`Bf:`	Blocs libres du planificateur, octets libres du tampon de réception.
`Ln:`	Le numéro de ligne du bloc en cours d'exécution.
`FS:`	La vitesse d'avance, la vitesse de broche demandée et la vitesse que rapporte le variateur. Une broche à simple marche/arrêt rapporte `F:`. Pendant un arrêt d'avance, l'avance est de 0.
`Pn:`	Entrées actives : les fins de course par lettre d'axe, `P` palpeur, `D` porte, `H` arrêt d'avance, `S` départ cycle, `R` réinitialisation, `E` arrêt d'urgence.
`WCO:`	Le décalage d'origine : tous les quelques rapports, et après un changement.
`Ov:`	Les corrections d'avance, de rapide et de broche en % : tous les quelques rapports, et après un changement.
`A:`	Accessoires : `S` ou `C` broche tournant dans le sens horaire ou antihoraire, `F` arrosage, `M` pulvérisation.
`Sl:`	La charge du variateur de broche.
`DTG:`	La distance restante, jusqu'à l'arrêt des axes (`$10` bit 13).
`H:`	Prise d'origine faite ou non.
`T:`	L'outil, pendant un changement d'outil.
`TLR:`	Une longueur d'outil de référence est définie (`$TLR`).
`MPG:`	La manivelle a le contrôle (1) ou non (0).
`In:`	Le résultat du dernier `M66`.
`WCS:`, `Sc:`, `D:`	Le système de coordonnées pièce, la mise à l'échelle, le mode diamètre du tour (rapports complets).
`AR`, `AR:`	Rapports automatiques désactivés, ou activés avec leur intervalle (rapports complets).
`FW:`	Le nom du micrologiciel (rapports complets).
`SD:`	L'avancement d'un programme exécuté depuis le système de fichiers du contrôleur.

Autres lignes envoyées par le contrôleur

Ligne	Quand
`ALARM:N`	Une alarme. Codes d'alarme donne la liste de tous les N.
`[MSG:…]`	Un message : celui du contrôleur, ou le `(MSG,…)`, `(PRINT,…)` ou `(DEBUG,…)` d'un programme.
`[GC:…]`	L'état modal, après `$G` ou `0x83`. Avec `$10` bit 9, une fois après chaque changement.
`[PRB:x,y,z:1]`	Le point où le palpeur s'est déclenché, et s'il a touché (1) ou non (0).
`[TLO:…]`	La correction de longueur d'outil.
`[PARAM:n=value]`	Après `$#=n`.

Ligne	Quand
`[G54:…]` … `[G59.3:…]`, `[G28:…]`, `[G30:…]`, `[G92:…]`, `[T:…]`	Après `$#` : les décalages, les positions mémorisées, la table d'outils.
`>line:ok`, `>line:error:N`	Une ligne de démarrage (`$N0`, `$N1`) s'est exécutée au démarrage.

Commandes `$`

`$HELP Commands` les liste sur le contrôleur lui-même.

Commande	Effet
`$`, `$HELP`, `$HELP <group>`	L'aide ; les commandes ; un groupe de réglages avec détails.
`$$`, `$<n>`, `$<n>=<value>`, `$$=<n>`, `$+`	Tous les réglages ; un réglage ; le modifier ; les détails d'un réglage ; de nouveau tous les réglages.
`$SED=<n>`	Décrire un réglage.
`$#`, `$#=<n>`, `$#=<name>`	Les décalages, le résultat du palpage, les positions mémorisées ; un paramètre.
`$G`	L'état modal.
`$I`, `$I+`, `$I=<text>`	Ce qu'est le contrôleur (ci-dessus) ; tout ; enregistrer un texte qui l'accompagne.
`$N`, `$N0=<line>`, `$N1=<line>`	Afficher ou enregistrer les lignes de démarrage.
`$C`	Mode vérification : le G-code est vérifié et rien ne bouge. Une réinitialisation y met fin.
`$X`	Déverrouiller après une alarme.
`$H`, `$HX`, `$HY`, `$HZ`	Faire la prise d'origine de tous les axes, ou d'un seul.
`$J=<G-code>`	Déplacement manuel (jog).
`$B`, `$S`, `$O`	Saut de bloc, bloc à bloc, arrêt optionnel : activés ou désactivés.
`$SLP`	Veille.
`$TLR`, `$TPW`	Prendre le dernier palpage comme longueur d'outil de référence ; mesurer le nouvel outil sur la plaque de contact.
`$RST=$`, `$RST=#`, `$RST=*`	Les réglages d'usine de la MC-1 (prise d'origine et fins de course actives, ses courses) ; effacer les décalages ; les deux, plus les lignes de démarrage.
`$EA`, `$EAG`, `$EE`, `$EEG`, `$EG`, `$ES`, `$ESG`, `$ESH`, `$E*`	Lister les alarmes, les codes d'erreur, les groupes de réglages et les réglages, en lignes ou en CSV.
`$SPINDLES`, `$SPINDLESH`	Les broches.
`$PINS`, `$PINSTATE`, `$PORTS`	Les pins de la carte ; les entrées et sorties auxiliaires ; les ports série.
`$LIM`, `$HSS`, `$LEV`	Les entrées de fin de course, les capteurs d'origine, les derniers événements d'entrée.
`$RTC`, `$RTC=<date and time>`	L'horloge du contrôleur (ISO 8601).
`$REBOOT`	Redémarrer le contrôleur.
`$F`, `$F+`, `$F=<file>`, `$F<=<file>`, `$FR`	Lister les programmes du système de fichiers du contrôleur, en exécuter un, en renvoyer un, recommencer.
`$FD=`, `$FMD=`, `$FRD=`, `$FF=yes`, `$FI`, `$FM`, `$FU`, `$CWD=`, `$PWD`	Supprimer, créer ou supprimer un dossier, formater, informations, monter, démonter, dossier de travail.

Chaque réglage figure dans Réglages.

Formats de fichiers

Brouillon (itération 14) : le texte est en cours de rédaction ; les sections peuvent encore s'allonger.

Programmes

Un programme est un fichier texte G-code dans le dialecte de Grbl 1.1, que parle le noyau propre à Twinwright. Studio lit n'importe quel fichier de ce genre. Les programmes que sa FAO écrit ressemblent à ceci :

```
(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
```

Élément	Ce que c'est
En-tête	Des commentaires : le titre du travail ; le premier outil, la matière et les principales données de coupe ; l'emplacement du zéro du programme
Modes	`G21 G90 G17 G94 G54` : millimètres, absolu, plan XY, avance par minute, décalage d'origine G54
`G53 G0 Z…`	Un déplacement vers le haut de Z, en coordonnées machine : au début, à chaque changement d'outil et à la fin. Cette hauteur est celle où la prise d'origine laisse Z.
`T… M6`	Un changement d'outil. Le contrôleur mesure le nouvel outil sur le capteur de longueur d'outil.
`M3 S…`, `G4 P…`	La broche en marche, puis une temporisation pendant qu'elle monte en vitesse. La pompe d'une broche refroidie par eau (`M8`) passe en premier.
Commentaires d'entité	Chaque entité commence par un commentaire qui dit ce qu'elle usine
Déplacements	`G0`, `G1`, et les arcs `G2`/`G3` avec `I` et `J`. Le perçage utilise `G81`, ou `G83` avec débourrage, son plan `R` à 1 mm au-dessus de la face supérieure.
Fin	`M5`, `M9`, le déplacement vers le haut, `M30`

Les commentaires sont entre parenthèses. Les lignes ne sont pas numérotées.

Studio conserve chaque version d'un programme dans le dossier des programmes (Documents/Twinwright Studio/programs), dans un dossier portant le nom du programme :

Fichier	Ce qu'il contient
`v0001.nc`	Le texte du programme, tel qu'il a été enregistré
`v0001.json`	Sa version, l'heure (UTC), qui l'a créé, pourquoi, la version dont il provient, son hachage et ses lignes
`v0001.ops.json`	Ce que chaque entité est devenue (opération, stratégie, outil, lignes, vitesse, avance) : une fiche de bridage le lit

Travaux et phases

Un travail est la description d'une pièce à partir de laquelle la FAO produit des programmes. C'est un objet JSON :

Clé	Signification
`name`	Le nom du programme, par exemple `bracket.nc`
`title`, `description`	Les textes de l'en-tête du programme et de la fiche de bridage
`material`	La matière, par exemple `6061-T6` ; les désignations d'autres normes sont comprises
`tool`	L'outil dans la broche, sous forme de spécification (`flat 6 n3 L20`)
`tools`	La table d'outils : `{"1": "flat 6 n3 L20", "2": "drill 3.4 118 L20"}`. Si elle est omise : celle de la machine.
`cutting`	Des valeurs qui remplacent les valeurs choisies : `rpm`, `feed`, `ap`, `ae`, `finish_ap` ; `corner_steps` (true ou false : les angles intérieurs d'une paroi taillés d'abord par paliers)
`safe_z`	La hauteur à laquelle se font les déplacements de transit, en millimètres au-dessus de l'origine pièce
`zero`	L'emplacement de l'origine pièce : `front-left-top` (par défaut) ou `center-top`
`features`	Les entités, dans l'ordre où elles sont usinées. Voir Créer des programmes.

Clé	Signification
`setups`	À la place de `features` : un objet par phase
`nest`	À la place de `features` : des pièces imbriquées sur une plaque

Une phase a ses propres `features` et indique comment la pièce y arrive depuis la phase précédente :

Clé	Signification
`turn`	`u` : retournée autour de X (l'avant et l'arrière changent de place) ; `v` : autour de Y (la gauche et la droite)
`fixture`	Maintenue autrement à partir de cette phase (`vise`, `clamps`, `tape` …)
`zero`	L'origine pièce retrouvée : `front-left-top` ou `center-top`. Si elle est omise après un retournement ou un nouveau montage : `front-left-top` ; sinon, l'origine de la phase précédente.
`name`, `tool`, `tools`, `cutting`, `safe_z`	Ceux propres à la phase, prioritaires sur ceux du travail

```
{"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}]}
```

Chaque version d'un travail est conservée à côté du programme qu'elle a produit, sous le nom `job-v0001.json` : sa version, l'heure (UTC), qui l'a créée (`operator` ou `ai:<client>`), une note, la version du programme qu'elle a produit et le travail lui-même. Un travail décrit en phases liste aussi, pour chaque phase, son programme, la version produite et son hachage.

Enregistrements (MCAP)

`--mcap FILE` enregistre une exécution dans un fichier MCAP (mcap.dev, format version 0) : un canal pour chaque type de données, des messages JSON, chaque canal avec son schéma JSON, sans découpage en blocs et sans compression, et un résumé à la fin avec les statistiques et un CRC. Les tests de Studio relisent les fichiers selon la spécification MCAP ; les outils qui lisent MCAP, comme PlotJuggler et Foxglove, sont censés les ouvrir, mais cela n'a pas encore été essayé.

- `--rec-period MS` règle la fréquence d'écriture des canaux échantillonnés (1 ms si l'option n'est pas donnée).
- Un enregistrement ne change rien à l'exécution : une exécution enregistrée et la même exécution non enregistrée sont identiques bit à bit.
- `--play FILE.mcap` relit un enregistrement dans Studio, avec une ligne de temps : lecture, pause, positionnement, de 0,25 à 10 fois la vitesse. Rien n'est simulé pendant la relecture.

Temps

L'horodatage d'enregistrement de chaque message est le temps virtuel du jumeau, en nanosecondes depuis la mise sous tension. La plupart des canaux portent aussi `t`, le même temps en secondes. Un canal échantillonné n'écrit un message que lorsque quelque chose y a changé.

L'en-tête

Un enregistrement de métadonnées nommé `mc1sim` décrit l'exécution :

Clé	Signification
`machine`	Le nom de la machine
`drive_model`	Le modèle de variateur de l'exécution : `L0`, `L1`, `L2`, `L2/3` (trois masses) ou `L3`
`f_cpu`	L'horloge du contrôleur, en Hz
`nvs`	Un hachage des réglages mémorisés du contrôleur au début de l'exécution, 16 chiffres hexadécimaux
`sample_period_ms`	La période d'échantillonnage
`time`	`virtual, ns from power-on`

Canaux

Canal	Écriture	Champ	Type	Unité	Signification
`/machine`	échantillonné	`pos`	nombre[3]	mm	Où se trouve chaque axe : la mécanique, non la position que le contrôleur s'en fait
		`mpos`	nombre[3] ou null	mm	La position machine du contrôleur (MPos) ; null tant que le contrôleur est éteint
		`lost`	nombre[3]		Les pas perdus par chaque axe
		`feed`	nombre	mm/min	La vitesse d'avance du contrôleur à l'instant présent

Canal	Écriture	Champ	Type	Unité	Signification
		`tip`	nombre[3]	mm	La pointe de l'outil par rapport à la pièce : avec le modèle d'erreurs du jumeau activé, déplacée par ses erreurs ; désactivé, identique à `pos`
`/drives`	échantillonné, avec un modèle de variateur à partir de L1	`ferr_um`	nombre[3]	µm	L'erreur de poursuite de chaque variateur
		`state`	chaîne[3]		L'état de chaque variateur
		`km1`	booléen		Le contacteur KM1 est fermé
		`bus48`	booléen		L'alimentation 48 V des variateurs est active
`/spindle`	échantillonné, avec un variateur de broche	`f_out`	nombre	Hz	La fréquence de sortie du variateur
		`rpm`	nombre	tr/min	La vitesse de la broche
		`amps`	nombre	A	Le courant de sortie du variateur
		`load_pct`	nombre	%	La charge telle que le variateur la rapporte
		`torque_nm`	nombre	N·m	Le couple du moteur
		`bus_v`	nombre	V	La tension du bus continu du variateur
		`cmd_rpm`	nombre	tr/min	La vitesse que demande le contrôleur
		`run`, `pump`	booléen		Le variateur en marche ; la pompe en marche
		`power`	chaîne		L'alimentation propre du variateur : `ready`, `boot` ou `off`
		`fault`	chaîne		Le défaut du variateur, tel que son afficheur l'indique
`/controller`	à chaque changement	`state`	chaîne		L'état du contrôleur (`Off` tant qu'il est éteint)
		`alarm`	entier		L'alarme dans laquelle il se trouve (0 : aucune)
		`homed`	entier		Les axes dont la prise d'origine est faite, un bit chacun : X 1, Y 2, Z 4
		`limits`	entier		Les entrées de fin de course qui sont actives, un bit chacune
		`probe`, `estop`, `motor_fault`, `door`	booléen		Les entrées du palpeur, de l'arrêt d'urgence, du défaut moteur et de la porte (la MC-1 n'a pas de porte)
		`on`	booléen		Le contrôleur est sous tension
`/power`	à chaque changement	`mains`, `v24`, `mcu`, `km1`, `bus48`	booléen		Le secteur ; l'alimentation 24 V ; le contrôleur en marche ; KM1 ; l'alimentation 48 V
		`vfd_mains`, `vfd`, `pump`	booléen, chaîne, booléen		Le secteur du variateur de broche ; son état (`none`, `ready`, `boot`, `off`) ; la pompe
`/serial/in`	à l'arrivée	`text`	chaîne		Les octets que le contrôleur a reçus ; les caractères de contrôle sous forme d'échappements `\u`
`/serial/out`	à l'arrivée	`line`	chaîne		Chaque ligne que le contrôleur a écrite
`/events`	au fil des événements	`text`	chaîne		Chaque entrée du journal d'événements : repères, alimentation, variateurs, variateur de broche, collisions, fils rompus
`/cut`	échantillonné, avec un brut	`cutting`	booléen		De la matière est enlevée ; sinon, les valeurs ci-dessous valent 0, sauf `rpm` et `removed_mm3`
		`mrr_mm3_s`	nombre	mm^3/s	Le débit d'enlèvement de matière
		`ap_mm`, `ae_mm`	nombre	mm	La profondeur axiale et la largeur radiale de la coupe
		`fz_mm`	nombre	mm	L'avance par dent
		`rpm`	nombre	tr/min	La vitesse de la broche
		`fx_n`, `fy_n`, `fz_n`	nombre	N	L'effort de coupe selon les axes de la machine
		`torque_nm`, `power_w`	nombre	N·m, W	Le couple et la puissance de coupe
		`kc_j_mm3`	nombre	J/mm^3	L'énergie spécifique de coupe
		`removed_mm3`	nombre	mm^3	La matière enlevée jusqu'ici
`/input`	à l'arrivée	`line`	chaîne		Les entrées de la session, ligne par ligne telles que `--record` les écrit (voir Journaux de session)
`/shadow`	à l'arrivée, en mode ombre	`host_s`	nombre	s	Le temps de l'ordinateur
		`dir`	chaîne		`t` vers la machine réelle, `r` depuis elle
		`data`	chaîne		Les octets, chacun sous forme d'un caractère U+0000–U+00FF, jusqu'à 1 Ko par message

Canal	Écriture	Champ	Type	Unité	Signification
`/ui/state`, `/sound`	pour le lecteur				La trame d'état des vues (30 par seconde tant qu'elle change) et la trame du son (toutes les 10 ms de temps virtuel quand elle change). Propres à Studio : leurs champs peuvent changer d'une version à l'autre.

Traces de pins (VCD)

`--vcd FILE` écrit les pins du contrôleur dans un fichier VCD (IEEE 1364), que lisent GTKWave et PulseView : le scope `mcu` contient les pins du contrôleur, chaque changement au tick d'horloge où il se produit (les pins de pas sous forme d'impulsions de `$0` microsecondes) ; le scope `machine` contient les lignes extérieures au contrôleur : les alimentations, KM1, le variateur de broche, la pompe, le bouton d'arrêt d'urgence, le capteur de longueur d'outil, les cinq capteurs. L'échelle de temps est de 1 ns.

Journaux de session

`--record FILE` écrit les entrées d'une session, chacune avec son temps virtuel : ce qu'ont envoyé les logiciels d'envoi, les commandes temps réel, les commandes des vues. `--replay FILE` rejoue la session, bit à bit, avec la même version de Studio et les mêmes fichiers machine. `--replay FILE --branch S` la rejoue jusqu'à S secondes, puis continue en direct à partir de là ; avec `--record`, la branche a un journal à elle, qui part de celui de son parent (`# branch <parent> <ticks>`). Un journal est un texte ; ses lignes sont propres à Studio et peuvent changer d'une version à l'autre.

L'archive du monde

Ce que le jumeau conserve entre deux sessions, comme le fait une machine réelle quand on l'éteint puis la rallume : la position des axes, le brut avec ses usinages, l'outil dans la broche, la mémoire du contrôleur et l'horloge de la carte, et la chaleur, qui retombe pendant que la machine est éteinte. Le contrôleur démarre à froid : la prise d'origine de la machine est à refaire.

La fenêtre conserve le monde de chaque machine dans le dossier d'état (sous Windows `%LOCALAPPDATA%\Twinwright\Studio`) : celui de la première machine dans `world`, celui des autres dans `machines\<id>\world`. `--world DIR` utilise un autre dossier ; `--world-reset` repart de l'état d'usine. Les exécutions par lots, les tests, les rejeux, les enregistrements relus et les machines d'arrière-plan de l'IA ne conservent aucun monde si `--world` n'est pas donné. Un seul processus à la fois ouvre un monde ; un deuxième repart de l'état d'usine et n'enregistre rien.

Fichier	Ce qu'il contient
`world.json`	La fin de la session (ci-dessous), écrite d'abord dans `world.json.tmp`, puis renommée
`stock.snap`	Les cellules du brut, usinées ou non (binaire, `TWSTOCK1`)
`eeprom.bin`	La mémoire conservée du contrôleur : ses réglages, ses décalages et ses données d'outils, écrits au fil de leurs changements

`world.json` :

Clé	Signification
`world`	La version du format : 1
`saved_at`, `saved_epoch`	Quand il a été enregistré : en UTC, et en secondes depuis 1970
`version`	La version de Studio qui l'a enregistré
`pos`	Où se trouvaient les axes, en mm (X, Y, Z)
`tool`	L'outil dans la broche, sous forme de spécification (`flat 6 n3 L20`)
`tool_table`	La table d'outils du travail : `{"1": "flat 6 n3 L20", ...}`
`tool_auto`	Le jumeau joue l'opérateur lors d'un changement d'outil : il met en place l'outil de la table et appuie sur départ cycle
`setup`	Le bridage et le brut, tels que les écrit le bouton Enregistrer le bridage de la fenêtre (null : aucun)
`cells`	`stock.snap` contient les cellules du brut
`thermal`	La chaleur : `screws` (chaque vis au-dessus du bâti), `frame` (le bâti au-dessus de 20 °C), `ambient` (la salle au-dessus de 20 °C), `part_heat` et `part_offset` (la pièce au-dessus du bâti : ce que les coupes y apportent, et un décalage donné), tous en K ; `spindle_um`, la croissance de la broche en µm
`rtc`	La lecture de l'horloge de la carte, en s
`probe_cal`	Le palpeur tel qu'il a été étalonné en dernier : le diamètre de la bille, son rayon effectif, les lobes triples et leur phase, le diamètre de la bague étalon, l'écart maximal, quand et par qui (seulement une fois qu'il a été étalonné)
`odometer`	`travel_mm` et `reversals` de chaque axe, `spindle_h` (les heures de la broche) et `on_h` (les heures du logiciel), cumulés sur les sessions

Au démarrage suivant, les axes sont là où ils se trouvaient ; le même brut revient avec ses cellules, ainsi que l'outil et la table ; la chaleur a diminué pendant le temps où la machine était éteinte, chaque élément vers son repos selon sa propre constante de temps ; l'horloge de la carte a continué de tourner. `--world-off S` indique combien de temps la machine est restée éteinte (pour les tests ; sinon, le temps écoulé depuis l'enregistrement). `--pos`, `--tool`, `--stock`, `--setup` et `--part` donnés au démarrage l'emportent sur l'archive.

Fichiers de licence

Un fichier de licence (`.twlic`) est un objet JSON en UTF-8, signé avec Ed25519. Studio le vérifie sur l'ordinateur, sans le réseau. Où se trouvent les fichiers et comment ils sont utilisés : Licences, postes et activation.

Cet exemple provient des vecteurs de test partagés. Il est signé avec la clé de test (`kid test`), que les versions publiées n'acceptent pas :

```
{"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"}
```

Champs

Champ	Type	Signification
`v`	entier	La version du format : 1. Studio considère comme non valide un fichier d'une autre version.
`kid`	chaîne	Laquelle des clés de Twinwright a signé le fichier
`id`	chaîne	Le numéro de la licence : `TWL-` suivi de 10 lettres et chiffres. Le renouvellement l'envoie.
`edition`	chaîne	`maker` ou `pro`
`kind`	chaîne	`subscription`, `perpetual` (achat définitif) ou `owner` (propriétaire d'une MC-1)
`seats`	entier	Le nombre de postes
`holder`	chaîne	Le titulaire de la licence : une personne ou une organisation. Jamais une adresse e-mail.
`issued`	date et heure	Quand le fichier a été signé, en UTC, `YYYY-MM-DDTHH:MM:SSZ`
`valid_until`	date et heure, ou null	Abonnement : la fin de la période payée. Achat définitif : la fin de la période de remboursement dans le premier fichier, null dans le fichier permanent.
`renew_after`	date et heure, ou null	À partir de quand Studio essaie de renouveler le fichier
`updates_until`	date, ou null	Achat définitif, propriétaire d'une MC-1 : les versions publiées jusqu'à ce jour ont les fonctions payantes
`fp`	tableau de chaînes	Les ordinateurs auxquels la licence est liée : les hachages de leurs empreintes, jusqu'à trois par poste
`mc1_serial`	chaîne, ou null	Propriétaire d'une MC-1 : le numéro de série de la machine. Studio ne vérifie pas la machine lorsqu'il s'exécute.
`sig`	chaîne	La signature Ed25519 des autres champs, en base64url sans remplissage

Un champ que Studio ne connaît pas fait partie des octets signés, et Studio ignore ce qu'il signifie.

Les octets signés

La signature couvre le fichier sans `sig`, écrit sous une forme canonique :

- les clés dans l'ordre des points de code Unicode ;
- aucun espace blanc ;
- des chaînes avec les échappements les plus courts de JSON ;
- des entiers sans point décimal ; aucun autre nombre, et aucune clé en double.

Studio écrit le fichier sous cette forme avant de vérifier la signature. Les espaces, les sauts de ligne et l'ordre des clés dans le fichier n'ont donc pas d'importance. Une valeur modifiée fait échouer la vérification.

Le hachage de l'empreinte

Le hachage est formé des 32 premiers chiffres hexadécimaux du SHA-256 calculé sur les octets `twinwright-fp-v1`, un octet nul, et l'identifiant de l'ordinateur :

Système	Identifiant de l'ordinateur
Windows	La valeur `MachineGuid` sous `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography`
Linux	`/etc/machine-id`

À défaut de l'un comme de l'autre, Studio crée un identifiant aléatoire et le conserve dans le dossier des licences (`machine.id`) ; le supprimer fait de l'ordinateur un autre ordinateur. L'identifiant lui-même ne quitte jamais l'ordinateur.

Par exemple, l'identifiant `8b1c3a0e-5f2d-4c7a-9e61-2d0f4b7a1c55` donne le hachage `0a2d31bb82176516415efa7b3a340562`.

Le code de demande

Le code de demande pour l'activation hors ligne est le JSON canonique en UTF-8 `{"fp": <the hash>, "version": <Studio's version>}`, en base64url sans remplissage `=`. Par exemple, le hachage `0123456789abcdef0123456789abcdef` et la version 1.0.0 donnent :

```
eyJmcCI6IjAxMjM0NTY3ODlhYmNkZWYwMTIzNDU2Nzg5YWJjZGVmIiwidmVyc2lvbiI6IjEuMC4wIn0
```

`TwinwrightStudio.exe --licence-request` affiche le code de demande de cet ordinateur.

Fichiers d'essai

Un essai est un fichier de licence comme les autres, signé par le serveur de licences de Twinwright : `"kind": "trial"`, le hachage de cet ordinateur dans `fp`, et `valid_until` 30 jours après le début de l'essai. Il n'est jamais renouvelé. Un fichier d'essai modifié à la main n'est plus valable.

L'archive de la machine

Chaque clé de l'archive d'une machine, avec les valeurs de la MC-1 : Schéma du profil de machine.

Logiciels d'envoi avec le noyau propre à Twinwright

Cette page s'adresse à ceux qui veulent piloter la MC-1 depuis un logiciel d'envoi de G-code de leur choix (bCNC, gSender ou Universal G-code Sender) plutôt que depuis Studio. Le contrôleur de la MC-1 est le noyau propre à Twinwright, compatible avec le protocole série Grbl 1.1 et les extensions qu'utilisent les logiciels d'envoi actuels ; ces logiciels d'envoi fonctionnent donc avec lui, moyennant les quelques réglages et réserves ci-dessous.

Tout ce qui figure ici a été testé sur le jumeau numérique de la MC-1, qui exécute le même code de contrôleur que la machine. La vérification sur une machine réelle reste à faire. Versions testées : bCNC 0.9.16, gSender 1.6.4, UGS Platform 2.1.27.

Connexion

Où	Comment
Le jumeau dans la fenêtre de Studio	Studio accepte un seul logiciel d'envoi à la fois sur `localhost:2323` (cet ordinateur seulement ; lancez Studio avec `--bind any` pour y accéder depuis le réseau). bCNC : port `socket://localhost:2323`. gSender : Ethernet, adresse `127.0.0.1`, port `2323`. UGS : `tcp://localhost:2323`.
La MC-1 elle-même	USB natif : le contrôleur apparaît comme un port série (COM*n* sous Windows). Choisissez ce port. Le débit en bauds est sans importance en USB natif, mais certains logiciels d'envoi en demandent un : 115 200 convient.

Ce que le contrôleur dit de lui-même :

- Sa ligne d'accueil est `Grbl 1.1f ['$' or '$HELP' for help]`, comme celle d'un contrôleur Grbl classique. Les logiciels d'envoi cherchent une ligne d'accueil qui commence par « Grbl » pour remarquer que le contrôleur a redémarré (voir la section 3).
- Son nom figure dans la réponse à `$I` : `[VER:Twinwright 1.1f.…]` et `[FIRMWARE:Twinwright]`. Les logiciels d'envoi qui affichent le micrologiciel affichent « Twinwright ».

Choisir le micrologiciel dans le logiciel d'envoi

Logiciel d'envoi	Réglage	Pourquoi
bCNC 0.9.16	Controller : **GRBL1**	bCNC ne lit pas le nom du micrologiciel. Testé avec GRBL1.
gSender 1.6.4	Default firmware : le choix autre que **Grbl**	gSender reconnaît un micrologiciel à son nom dans `$I`. Notre contrôleur indique son propre nom ; gSender utilise donc le micrologiciel que vous avez défini par défaut. Testé avec ce choix.
UGS 2.1.27	Firmware : **GRBL**	Testé : réglages, programmes, alarmes et prise d'origine après une réinitialisation, sans Soft Reset supplémentaire (section 3). L'autre choix de micrologiciel d'UGS fonctionne aussi, mais demande un Soft Reset de plus après une réinitialisation ; le déplacement manuel (jog) n'a été testé qu'avec ce choix.

Après une alarme ou une réinitialisation

- **Arrêt d'urgence.** L'arrêt d'urgence de la MC-1 ouvre le contacteur de puissance. Relâchez-le, appuyez sur le bouton Start de la machine, puis réinitialisez dans le logiciel d'envoi et faites la prise d'origine. Tant que Start n'est pas actionné, l'alarme (ALARM:10) ne s'efface pas.
- **ALARM:11 (prise d'origine requise).** Faites la prise d'origine (`$H`). Le déverrouillage avec `$X` est refusé (error:46).
- **ALARM:2 (limite logicielle).** Le programme a demandé un déplacement hors de la course. La prise d'origine laisse Z en haut de sa course (Z machine 0) ; tout déplacement au-dessus, comme `G53 G0 Z5`, déclenche donc ALARM:2. Réinitialisez, puis faites la prise d'origine ou déverrouillez.
- **bCNC** se rétablit après toute réinitialisation : il voit la ligne d'accueil « Grbl » et oublie les lignes qui étaient encore en route, de sorte que `$H` part bien.
- **bCNC après Stop.** Le Stop de bCNC est un arrêt d'avance suivi d'une réinitialisation. Ensuite, bCNC renvoie au contrôleur, en une ou deux lignes, les modes qu'il avait auparavant (système de coordonnées, unités, correction de longueur d'outil), et dans quelques cas le contrôleur les refuse, comme le font d'autres contrôleurs Grbl : avec une correction de longueur d'outil (`G43.1`) active, la ligne reçoit error:24 (bCNC envoie son propre `G43.1Z...` après son premier Stop ; à partir du deuxième Stop, cela arrive donc toujours) ; sous `G80`, `G43.1Z...` reçoit error:31 ; avec une mise à l'échelle (`G51`) active, `G51:X` reçoit error:1. Après une erreur, chaque ligne de G-code qui suit reçoit la même erreur jusqu'à une ligne vide, une commande `$` ou une réinitialisation. Après un Stop, vérifiez donc le système de coordonnées, les unités et la correction de longueur d'outil avec `$G` et `$#` avant de faire un déplacement manuel ou de continuer. Si ces valeurs sont fausses, redéfinissez l'origine pièce et mesurez de nouveau la longueur d'outil (testé sur le jumeau, 2026-10-06).
- **UGS avec son autre choix de micrologiciel.** Après une réinitialisation qu'UGS n'a pas envoyée lui-même, par exemple une réinitialisation depuis la fenêtre de Studio ou un redémarrage spontané du contrôleur, appuyez **une fois de plus sur Soft Reset** dans UGS avant Home. Tant que ce n'est pas fait, Home ne fait rien. Avec ce choix, UGS ne prend pour un redémarrage qu'une autre ligne d'accueil. Une réinitialisation par le bouton Soft Reset d'UGS lui-même ne pose pas de problème. Sur la MC-1 elle-même, de telles réinitialisations sont rares : elle n'a pas d'entrée de réinitialisation et un seul hôte USB. Avec le micrologiciel réglé sur « GRBL », cela ne se produit pas : après une telle réinitialisation, UGS affiche CONNECTING, et Home fonctionne aussitôt (testé sur le jumeau, 2026-10-06).

Changements d'outil (M6)

Aucun des trois logiciels d'envoi, avec ses réglages par défaut, ne permet à la MC-1 de changer l'outil et de mesurer le nouveau :

Logiciel d'envoi	Par défaut	Que faire
bCNC	Ignore chaque ligne M6 : les coupes de l'outil suivant sont faites avec l'outil resté dans la broche.	Réglez son changement d'outil sur **Manual tool change (TLO)** et mesurez sur le capteur de longueur d'outil de la MC-1 (section 5).

Logiciel d'envoi	Par défaut	Que faire
gSender	Changement d'outil « Ignore » : M6 est envoyé comme commentaire et le programme ne s'arrête pas.	Réglez son changement d'outil sur **Fixed Tool Sensor**, au niveau du capteur de longueur d'outil de la MC-1 (section 5).
UGS	Gère M6 lui-même : il dégage, arrête la broche et vous demande de changer l'outil, mais il ne le mesure jamais. Un outil plus long coupe alors trop profond et, sur une machine réelle, il peut entrer en collision avec la pièce.	Exécutez un seul outil par programme et définissez la longueur de l'outil ou l'origine pièce avant chacun, ou utilisez Studio pour les programmes avec changements d'outil.

Compléments sur UGS et M6 :

- Il s'arrête aussi à un « M6 » dans un commentaire, comme le `( M6 T1 )` de FreeCAD.
- Un deuxième M6 qui suit immédiatement un changement d'outil peut lui échapper et arriver jusqu'au contrôleur. Le contrôleur attend alors, à l'état Tool, une confirmation qu'UGS n'envoie jamais, et plus aucun bouton de la fenêtre d'UGS ne fonctionne. Sur la MC-1, seul l'arrêt d'urgence y met fin : appuyez dessus, relâchez-le, appuyez sur Start, puis sur Soft Reset et Home dans UGS.
- Si le contrôleur est réinitialisé pendant un travail UGS avant qu'UGS soit arrivé à un M6, UGS peut signaler le travail comme terminé, mais sa boîte de dialogue de changement d'outil s'affiche quand même. La fermer peut relancer tout le programme depuis le début. Après une réinitialisation en plein travail, fermez le fichier dans UGS avant de continuer.

Mesurer la longueur d'outil sur le capteur de longueur d'outil de la MC-1

Le capteur de longueur d'outil se trouve à l'arrière gauche de la table ; `G59.3` est mémorisé juste au-dessus.

- **bCNC, Manual tool change (TLO).** Position de changement d'outil au-dessus du capteur de longueur d'outil. Position de palpage `G59.3`. Distance de palpage 50 mm, recherche 200 mm/min, approche lente 25 mm/min (`$342`–`$344` de la MC-1), commande G38.2. Mesurez le premier outil avec « Calibrate » avant d'exécuter le programme. bCNC définit ensuite `G43.1` pour chaque nouvel outil.
- **gSender, Fixed Tool Sensor.** Emplacement du capteur `G59.3`. Mesurez l'outil initial dans l'assistant. gSender décale ensuite le Z pièce pour chaque nouvel outil (`G10 L20`) au lieu d'utiliser une correction de longueur d'outil. Il prend la longueur à l'endroit où l'axe s'est arrêté après le contact, et non au point de déclenchement que rapporte le contrôleur ; chaque changement peut donc être un peu décalé. Dans le jumeau, exécutez les changements d'outil à la vitesse 1× : gSender lit cette position après une courte pause fixe, et un jumeau qui tourne plus vite peut lui en donner une antérieure à l'arrêt.
- **Commencez à au moins 0,1 mm sous le haut de Z** (Z machine −0,1 ou plus bas) quand le changement d'outil commence. Les deux logiciels d'envoi reviennent à cette hauteur après la mesure. En partant tout en haut, la petite dispersion d'une mesure peut placer ce retour au-dessus de la limite logicielle, et le travail s'arrête avec ALARM:2 avant d'avoir usiné.

Définir l'origine pièce avec un palpeur

- **UGS** (son module de palpage) calcule l'origine pièce à partir des points de déclenchement que rapporte le contrôleur (`[PRB:]`). C'est la bonne méthode.
- **gSender** (palpeur 3D) et **bCNC** (sa page de palpage, puis ses boutons Z=0 / X / Y) définissent l'origine pièce là où les axes se sont arrêtés après le contact. La machine continue toujours un peu au-delà du déclenchement avant de s'arrêter, et d'autant plus loin que le contact final est rapide. Utilisez un contact final lent, ou définissez l'origine pièce avec les cycles de palpage de Studio, qui utilisent les points de déclenchement.
- **Le centre de trou d'UGS** (le Hole Center de son module de palpage) : à partir de l'endroit où vous laissez le palpeur, il palpe en -X, +X, -Y et +Y, chaque fois sur la moitié du diamètre de trou que vous saisissez, et place le centre à mi-chemin entre les points de déclenchement des contacts lents. Sa portée dépend de ce diamètre : le point de départ peut être éloigné du centre d'au plus (diamètre saisi - alésage réel) / 2 + le rayon de la bille. Si vous saisissez le diamètre exact de l'alésage, cela ne fait que le rayon de la bille (1 mm pour une bille de 2 mm) ; saisissez un diamètre plus grand de quelques mm (24 à 26 pour un alésage de 20 mm) pour tenir compte d'un placement du palpeur à l'œil.
- **Quand le centre de trou d'UGS n'atteint pas la paroi** (constat 192, comportement propre à UGS) : ce palpage se termine par ALARM:5 et UGS s'arrête. Mais il a défini X0 Y0 de G54 au point de départ avant de palper et, comme le cycle ne s'est pas terminé, G54 reste là. Après le déverrouillage, redéfinissez l'origine pièce avant d'usiner. Son cycle de palpage reste aussi à mi-chemin : déconnectez-vous puis reconnectez-vous avant de palper de nouveau. Il prend aussi pour un contact un point que le contrôleur rapporte comme non touché (`[PRB:...:0]`).
- Étalonnez d'abord le palpeur (cycles de palpage de Studio : une bague étalon). Un palpeur non étalonné lit chaque alésage trop grand et chaque bossage trop petit du double de sa pré-course.

Autres points à connaître

- **bCNC sur le réseau.** Sous Windows, bCNC ne lit une connexion TCP qu'environ toutes les 2 secondes ; son affichage peut donc avoir 2 secondes de retard. « Idle » dans bCNC ne signifie pas que la machine s'est arrêtée. Une connexion USB n'a pas ce délai.
- **gSender, M0.** Un M0 situé dans les 10 premières lignes d'un programme ne met pas en pause (gSender le transforme en commentaire). M1 met toujours en pause.
- **gSender, déplacement manuel continu.** Maintenir un bouton de déplacement manuel est refusé sur la MC-1 (error:15) : gSender demande un long déplacement manuel au-delà de la course. Déplacez les axes par pas. Sur certains contrôleurs Grbl, `$40=1` (limiter les commandes de déplacement manuel) arrête les longs déplacements aux limites logicielles au lieu de les refuser ; le contrôleur de la MC-1 ne tient pas encore compte de ce réglage et les refuse toujours (essayé avec gSender sur le jumeau, 2026-10-06).
- **UGS, lignes longues.** UGS n'envoie jamais une ligne de plus de 127 caractères : le travail s'arrête sans message. Gardez des lignes plus courtes (UGS supprime d'abord les commentaires).
- **UGS, M30.** UGS supprime M30. Un programme qui compte sur M30 pour arrêter la broche et le refroidissement les laisse en marche : terminez les programmes par M5 et M9 avant M30.
- **UGS, commentaires.** Si une ligne contient deux commentaires, UGS supprime aussi le code situé entre eux.
- **Déconnexion pendant un travail.** Le contrôleur continue d'exécuter les lignes qu'il a déjà. gSender et UGS n'envoient aucun arrêt quand vous vous déconnectez ; la machine continue donc, broche en rotation. Arrêtez le travail avant de vous déconnecter. bCNC fait un arrêt d'avance puis une réinitialisation avant de fermer la connexion.

- **Commentaires avec des caractères non ASCII** (comme le chinois, « Ø », « – ») : les trois logiciels d'envoi suppriment les commentaires avant l'envoi ; ces caractères n'atteignent donc jamais le contrôleur. Le contrôleur lui-même lit le texte UTF-8 dans les commentaires et dans `(MSG,...)` comme du texte, jamais comme une commande temps réel.