Expressions – Référence

**Unités**

**Remarque :** spécifiez les unités à utiliser dans le calcul de l’expression. Sinon, les unités définies dans le document sont utilisées.

Tapez une unité après la valeur, par exemple :

* « 2 mm » au lieu de « 2 »
* « 5,2 mm » au lieu de « 5,2 »
* « 1 po » au lieu de « 1 »

**Exemples d’expressions**

| **Exemples de relations entre les paramètres** | **Format des expressions** |
| --- | --- |
| Moitié du diamètre d’outil | diamètre\_outil / 2 |
| 80 % de la longueur de la dent | longueurDent\_outil \* 0,8 |
| Minimum entre le diamètre de l’outil et 12 mm | Math.min (diamètre\_outil, 12 mm) |
| Calcul de la zone plane d’un outil « torique » | diamètre\_outil - (2 \* rayonCoin\_outil) |
| Calcul de la hauteur de crête d’un outil hémisphérique (remplacez 0,666 par la hauteur de crête de votre choix) | Math.sqrt((diamètre\_outil/2)^2 - (diamètre\_outil/2 - 0,666)^2) /2 |
| Pas à pas en fonction de l’outil. S’il s’agit d’une fraise de surfaçage, utiliser 95 % du diamètre. Dans le cas contraire, utilisez 40 % du diamètre. | type\_outil == ’fraise de surfaçage’ ? diamètre\_outil \* 0,95 : diamètre\_outil \* 0,4 |

**Instructions conditionnelles – Format**

Condition ? what happens when True : what happens when False

Le tableau suivant présente des exemples de conditions et indique comment les ajouter à des expressions pour obtenir un résultat basé sur la condition.

| **Exemples de conditions** | **Conditions ajoutées à une expression** |
| --- | --- |
| diamètre\_outil > 5 | diamètre\_outil > 5 ? 2 mm : 1 mm |
| recouvrement == 1 | recouvrement == 1 ? 0,1 mm : 0,2 mm |
| tolérance <= 0,02 | tolérance <= 0,02 ? 1 mm : 3 mm |

Instructions conditionnelles avec plusieurs conditions

Pour ajouter plusieurs conditions à une instruction, imbriquez les conditions de la manière suivante :

**Exemple 1**

Si Condition1 est *True*, alors Condition2 est exécutée. Le résultat de Condition2 est alors utilisé.

Condition ? (Condition2 ? True2 : False2) : False

**Exemple 2**

si Condition1 est *False*, alors Condition2 est exécutée. Le résultat de Condition2 est alors utilisé.

Condition ? True : (Condition2 ? True2 : False2)

**Tableau des opérateurs conditionnels**

| **Condition/Opérateur** | **Signification** |
| --- | --- |
| valeur1 == valeur2 | Vérifie si les valeurs sont identiques. |
| valeur1 != valeur2 | Vérifie si les valeurs sont différentes. |
| valeur1 > valeur2 | Vérifie si la valeur 1 est supérieure à la valeur 2. |
| valeur1 < valeur2 | Vérifie si la valeur 1 est inférieure à la valeur 2. |
| valeur1 >= valeur2 | Vérifie si la valeur 1 est supérieure ou égale à la valeur 2. |
| valeur1 <= valeur2 | Vérifie si la valeur 1 est inférieure ou égale à la valeur 2. |

**Fonctions mathématiques**

| **Fonction** | **Entrée 1** | **Entrée 2** | **Description** |
| --- | --- | --- | --- |
| Math.maximum(entrée1, entrée2) | Nombre | Nombre | Renvoie la plus basse des deux valeurs d’entrée. |
| Math.minimum(entrée1, entrée2) | Nombre | Nombre | Renvoie la plus élevée des deux valeurs d’entrée. |
| Math.sqrt(entrée1) | Nombre | N/A | Renvoie la racine carrée de la valeur d’entrée. |
| Math.sin(entrée1) | Nombre (radians) | N/A | Renvoie le sinus de la valeur d’entrée en radians. |
| Math.asin(entrée1) | Nombre | N/A | Renvoie le sinus inverse de la valeur d’entrée en radians. |
| Fonction | Entrée 1 | Entrée 2 | Description |
| Math.cos(entrée1) | Nombre (radians) | N/A | Renvoie le cosinus de la valeur d’entrée en radians. |
| Math.acos(entrée1) | Nombre | N/A | Renvoie le cosinus inverse de la valeur d’entrée en radians. |
| Math.tan(entrée1) | Nombre (radians) | N/A | Renvoie la tangente de la valeur d’entrée en radians. |
| Math.atan(entrée1) | Nombre | N/A | Renvoie la tangente inverse de la valeur d’entrée en radians. |
| Math.floor(entrée1) | Nombre à arrondir | N/A | Arrondit la valeur d’entrée à l’entier inférieur le plus proche. |
| Math.floorto(entrée1, entrée2) | Nombre à arrondir | Décimales de l’arrondi (0,01 arrondit à 2 décimales). | Arrondit le nombre à la décimale inférieure spécifiée la plus proche. |
| Math.ceil(entrée1) | Nombre à arrondir | N/A | Arrondit le nombre à l’entier supérieur le plus proche. |
| Math.ceilto(entrée1, entrée2) | Nombre à arrondir | Décimales de l’arrondi (0,01 arrondit à 2 décimales). | Arrondit le nombre à la décimale supérieure spécifiée la plus proche. |
| Math.round(entrée1) | Nombre à arrondir | N/A | Arrondit le nombre spécifié. |
| Math.roundto(entrée1, entrée2) | Nombre à arrondir | Décimales de l’arrondi (0,01 arrondit à 2 décimales). | Arrondit le nombre à la décimale spécifiée la plus proche. |
| Math.roundToSignificant(entrée1, entrée2) | Nombre à arrondir | Nombre de chiffres requis | Arrondit le nombre au nombre de chiffres spécifiés. |
| Math.normalizeAngleDegrees(entrée1) | Nombre (degrés) | N/A | Renvoie la valeur en degrés de l’angle d’entrée dans son équivalent dans la plage 0-360. |
| Math.isnan(entrée1) | N’importe quelle valeur | N/A | Renvoie **True** si la valeur d’entrée n’est pas un nombre et **False** s’il s’agit d’un nombre |
| Math.abs(entrée1) | Nombre | N/A | Renvoie la valeur absolue du nombre. |

One way to identify variables is to hold the SHIFT key while mousing over a CAM entry field. The variable definitions associated with that entry field will then be shown in the pop-up tooltip.  
  
The variables for the CAM workspace are listed below:

* advancedMode=false; // Advanced Mode TRANSIENT
* allowRapidRetract=true; // Allow Rapid Retract
* alphaMode=false; // Alpha Mode TRANSIENT
* betaMode=false; // Beta Mode TRANSIENT
* bottom=bottomOffset; // Bottom
* bottomAbsolute=false; // Absolute Bottom TRANSIENT
* bottomHeight\_value; //Absolute Bottom Height
* bottomMode='from contour'; // Bottom Mode
* bottomOffset=0in; // Bottom Offset
* bottomRef=undefined; // Bottom Reference DISABLED
* chainingTolerance=0.0004in; // Chaining Tolerance
* clearanceHeight=retractHeight + clearanceHeightOffset; // Clearance Height
* clearanceHeightAbsolute=true; // Absolute Clearance Height TRANSIENT
* clearanceHeightMode='from retract height'; // Clearance Height Mode
* clearanceHeightOffset=0.4in; // Clearance Height Offset
* clearanceHeightRef=undefined; // Clearance Height Reference DISABLED
* context='operation'; // Context TRANSIENT
* contourTolerance=tolerance \* 0.5; // Contour Linearization Tolerance
* cuspHeight=0mm; // Worst-Case Cusp Height TRANSIENT
* direction='climb'; // Direction
* doMultipleDepths=false; // Multiple Depths
* explicitRampDiameter=false; // Manual Ramp Diameter
* fineStepdown=maximumStepdown; // Fine Stepdown DISABLED
* helicalRampDiameter=Math.max((tool\_diameter - 2 \* tool\_cornerRadius) \* 0.95; tool\_diameter \* 0.25); // Helical Ramp Diameter
* highFeedrate=Math.max(tool\_feedCutting; tool\_feedEntry; tool\_feedExit); // High Feedrate DISABLED
* highFeedrateMode='disabled'; // High Feedrate Mode
* holder\_attached=false; // Attached Holder TRANSIENT
* holder\_comment=''; // Holder Comment TRANSIENT
* holder\_description=''; // Holder Description TRANSIENT
* holder\_libraryName=''; // Holder Library TRANSIENT
* holder\_productId=''; // Holder Product ID TRANSIENT
* holder\_vendor=''; // Holder Vendor TRANSIENT
* leadRadius=tool\_diameter \* 0.1; // Horizontal Lead In/Out Radius
* liftHeight=.1in; // Lift Height
* loadDeviation=(optimalLoad<(tool\_diameter-optimalLoad\*0.20))?(optimalLoad\*0.10):(tool\_diameter-optimalLoad)/2; // Load Deviation
* machineCavities=true; // Machine Cavities
* maximumStepdown=Math.min(tool\_fluteLength; 10mm); // Maximum Roughing Stepdown DISABLED
* metric=false; // Metric TRANSIENT
* minimumRadius=tool\_diameter \* 0.1; // Minimum Cutting Radius
* minimumRampDiameter=helicalRampDiameter; // Minimum Ramp Diameter
* minimumStayDownClearance=Math.max(Math.min(tool\_diameter \* 0.1; 15.0); 2.0); // Minimum Stay-Down Clearance
* noEngagementFeedrate=Math.max(tool\_feedCutting; tool\_feedEntry; tool\_feedExit); // No-Engagement Feedrate
* optimalLoad=tool\_diameter \* 0.45; // Optimal Load
* orderByDepth=false; // Order by Depth
* overrideToolView=false; // Tool Orientation
* rampAngle=4deg; // Ramping Angle (deg)
* rampClearanceHeight=0.1in; // Ramp Clearance Height
* rampDiameter=(tool\_diameter - 2 \* tool\_cornerRadius) \* 0.95; // Minimum Ramp Diameter DISABLED
* rampTaperAngle=0.0deg; // Ramp Taper Angle (deg)
* rampType='helix'; // Ramp Type
* reduceOnlyInnerCorners=true; // Only Inner Corners DISABLED
* reducedFeedChange=25deg; // Maximum Directional Change DISABLED
* reducedFeedDistance=tool\_diameter \* 0.2; // Reduced Feed Distance DISABLED
* reducedFeedRadius=tool\_diameter \* 0.05; // Reduced Feed Radius DISABLED
* reducedFeedrate=tool\_feedCutting \* 0.25; // Reduced Feedrate DISABLED
* retractHeight=stockZHigh + retractHeightOffset; // Retract Height
* retractHeightAbsolute=true; // Absolute Retract Heigth TRANSIENT
* retractHeightMode='from stock top'; // Retract Height Mode
* retractHeightOffset=0.2in; // Retract Height Offset
* retractHeightRef=undefined; // Retract Height Reference DISABLED
* roundWalls=false; // Round Walls DISABLED
* simpleStockToLeave=false; // Simple Stock To Leave
* slopeAngle=tool\_taperAngle; // Wall Taper Angle (deg) DISABLED
* slotClearingWidth=1.25 \* tool\_diameter; // Slot Clearing Width DISABLED
* smoothingFilter=false; // Smoothing
* smoothingFilterTolerance=tolerance \* 0.1; // Smoothing Tolerance DISABLED
* stayDownDistance=5 \* tool\_diameter; // Maximum Stay-Down Distance
* stayDownLevel='level10'; // Stay-Down Level
* stockBoundaryMode='bounding-box'; // Stock Boundary
* stockToLeave=0.005in; // Radial Stock to Leave
* stockZHigh=0.0089535m; // Top of the Stock TRANSIENT
* stockZLow=-0.0089535m; // Bottom of the Stock TRANSIENT
* strategy='adaptive2d'; // Strategy TRANSIENT
* surfaceTolerance=tolerance \* 0.5; // Surface Triangulation Tolerance
* surfaceZHigh=0.0079375m; // Top of the Geometry TRANSIENT
* surfaceZLow=-0.0079375m; // Bottom of the Geometry TRANSIENT
* tolerance=0.004in; // Tolerance
* toolViewOriginSelectionMode='setupWCS'; // Origin Selection Mode DISABLED
* tool\_bodyLength=(tool\_type == 'form mill') ? (tool\_overallLength) : (tool\_shoulderLength + tool\_diameter \* 2); // Body Length TRANSIENT
* tool\_breakControl=false; // Break Control TRANSIENT
* tool\_clockwise=tool\_type != 'tap left hand'; // Spindle Rotation TRANSIENT
* tool\_comment=''; // Comment TRANSIENT
* tool\_compensationOffset=tool\_number; // Compensation Offset TRANSIENT
* tool\_coolant='flood'; // Coolant
* tool\_cornerRadius=tool\_isTurning ? 0.8mm : tool\_diameter/4; // Corner Radius DISABLED TRANSIENT
* tool\_description=''; // Description TRANSIENT
* tool\_diameter=0.4in; // Diameter TRANSIENT
* tool\_diameterOffset=tool\_number; // Diameter Offset TRANSIENT
* tool\_feedCutting=40inpm; // Cutting Feedrate
* tool\_feedEntry=tool\_feedCutting; // Lead-In Feedrate
* tool\_feedExit=tool\_feedCutting; // Lead-Out Feedrate
* tool\_feedPerRevolution=tool\_feedPlunge/tool\_spindleSpeed; // Feed per Revolution
* tool\_feedPerTooth=tool\_feedCutting/(tool\_spindleSpeed \* tool\_numberOfFlutes); // Feed per Tooth
* tool\_feedPlunge=(tool\_type=='drill')?(40inpm):(tool\_feedCutting/3); // Plunge Feedrate
* tool\_feedRamp=tool\_feedPlunge; // Ramp Feedrate
* tool\_feedRetract=tool\_feedPlunge; // Retract Feedrate DISABLED
* tool\_fluteLength=(tool\_type=='drill')?(tool\_diameter \* 10):(tool\_diameter \* 2); // Flute Length TRANSIENT
* tool\_isDrill=false; // Tool is for drilling TRANSIENT
* tool\_isMill=true; // Tool is for milling TRANSIENT
* tool\_isTurning=false; // Tool is for turning TRANSIENT
* tool\_lengthOffset=tool\_number; // Length Offset TRANSIENT
* tool\_manualToolChange=false; // Manual Tool Change TRANSIENT
* tool\_material='hss'; // Material TRANSIENT
* tool\_number=1; // Number TRANSIENT
* tool\_numberOfFlutes=3; // Number of Flutes TRANSIENT
* tool\_overallLength=(tool\_isTurning || tool\_type == 'form mill') ? 30.0mm : tool\_bodyLength + tool\_diameter \* 2; // Overall Length TRANSIENT
* tool\_productId=''; // Product ID TRANSIENT
* tool\_rampFeedPerTooth=tool\_feedRamp/(tool\_rampSpindleSpeed \* tool\_numberOfFlutes); // Ramp Feed per Tooth
* tool\_rampSpindleSpeed=tool\_spindleSpeed; // Ramp Spindle Speed
* tool\_rampSurfaceSpeed=tool\_isTurning ? (200m/min) : (tool\_stockDiameter \* Math.PI \* tool\_rampSpindleSpeed); // Ramp Surface Speed
* tool\_shaftDiameter=tool\_diameter; // Shaft Diameter TRANSIENT
* tool\_shoulderLength=tool\_fluteLength + tool\_diameter; // Shoulder Length TRANSIENT
* tool\_spindleSpeed=tool\_isTurning ? 500rpm : 5000rpm; // Spindle Speed
* tool\_stockDiameter=tool\_isTurning ? (tool\_unit == 'millimeters' ? 25 : 25.4) : tool\_diameter; // Stock Diameter DISABLED
* tool\_surfaceSpeed=tool\_isTurning ? (200m/min) : (tool\_stockDiameter \* Math.PI \* tool\_spindleSpeed); // Surface Speed
* tool\_taperAngle=(tool\_type == 'chamfer mill') ? 45.0deg : 5.0deg; // Taper Angle DISABLED TRANSIENT
* tool\_threadPitch=0.04in; // Thread Pitch DISABLED TRANSIENT
* tool\_tipAngle=(tool\_type == 'counter sink') ? 90.0deg : 118.0deg; // Tip Angle DISABLED TRANSIENT
* tool\_tipDiameter=tool\_type == 'spot drill' ? 0.0 : tool\_diameter\*0.5; // Tip Diameter DISABLED TRANSIENT
* tool\_tipLength=(tool\_type == 'center drill')?(tool\_tipDiameter \* 2):(0); // Tip Length DISABLED TRANSIENT
* tool\_tipOffset=0; // Tip Offset DISABLED TRANSIENT
* tool\_turret=0; // Turret TRANSIENT
* tool\_type='flat end mill'; // Type TRANSIENT
* tool\_unit='inches'; // Unit TRANSIENT
* tool\_vendor=''; // Vendor TRANSIENT
* top=stockZHigh + topOffset; // Top
* topAbsolute=true; // Absolute Top TRANSIENT
* topHeight\_value; //Absolute Top Height
* topMode='from stock top'; // Top Mode
* topOffset=0in; // Top Offset
* topRef=undefined; // Top Reference DISABLED
* useEvenStepdowns=false; // Use Even Stepdowns DISABLED
* useFeedOptimization=false; // Feed Optimization
* useSlotClearing=false; // Use Slot Clearing
* useStockToLeave=true; // Stock to Leave
* verticalLeadRadius=leadRadius; // Vertical Lead In/Out Radius
* verticalStockToLeave=stockToLeave; // Axial Stock to Leave
* viewOrigin=''; // Tool Orientation Origin DISABLED
* viewPlane=''; // Tool Orientation Plane DISABLED
* viewReverseX=false; // Reverse X Axis DISABLED
* viewReverseY=false; // Reverse Y Axis DISABLED
* viewReverseZ=false; // Reverse Z Axis DISABLED
* viewSelection='originAndOrientation'; // Tool Orientation Selection DISABLED
* wcsOriginSelection='top center'; // WCS Origin DISABLED