
Autodesk® PowerMill® 2017

User Guide

Toolpath Point Parameters



Autodesk® PowerMill® 2017

© 2016 Delcam Limited. All Rights Reserved. Except where otherwise permitted by Delcam Limited, this publication, or parts thereof, may not be reproduced in any form, by any method, for any purpose.

Certain materials included in this publication are reprinted with the permission of the copyright holder.

Trademarks

The following are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and other countries: 123D, 3ds Max, Alias, ArtCAM, ATC, AutoCAD LT, AutoCAD, Autodesk, the Autodesk logo, Autodesk 123D, Autodesk Homestyler, Autodesk Inventor, Autodesk MapGuide, Autodesk Streamline, AutoLISP, AutoSketch, AutoSnap, AutoTrack, Backburner, Backdraft, Beast, BIM 360, Burn, Buzzsaw, CADmep, CAiCE, CAMduct, Civil 3D, Combustion, Communication Specification, Configurator 360, Constructware, Content Explorer, Creative Bridge, Dancing Baby (image), DesignCenter, DesignKids, DesignStudio, Discreet, DWF, DWG, DWG (design/logo), DWG Extreme, DWG TrueConvert, DWG TrueView, DWGX, DXF, Ecotect, Ember, ESTmep, FABmep, Face Robot, FBX, FeatureCAM, Fempro, Fire, Flame, Flare, Flint, ForceEffect, FormIt 360, Freewheel, Fusion 360, Glue, Green Building Studio, Heidi, Homestyler, HumanIK, i-drop, ImageModeler, Incinerator, Inferno, InfraWorks, Instructables, Instructables (stylized robot design/logo), Inventor, Inventor HSM, Inventor LT, Lustre, Maya, Maya LT, MIMI, Mockup 360, Moldflow Plastics Advisers, Moldflow Plastics Insight, Moldflow, Moondust, MotionBuilder, Movimento, MPA (design/logo), MPA, MPI (design/logo), MPX (design/logo), MPX, Mudbox, Navisworks, ObjectARX, ObjectDBX, Opticore, P9, PartMaker, Pier 9, Pixlr, Pixlr-o-matic, PowerInspect, PowerMill, PowerShape, Productstream, Publisher 360, RasterDWG, RealDWG, ReCap, ReCap 360, Remote, Revit LT, Revit, RiverCAD, Robot, Scaleform, Showcase, Showcase 360, SketchBook, Smoke, Socialcam, Softimage, Spark & Design, Spark Logo, Sparks, SteeringWheels, Stitcher, Stone, StormNET, TinkerBox, Tinkercad, Tinkerplay, ToolClip, Topobase, Toxik, TrustedDWG, T-Splines, ViewCube, Visual LISP, Visual, VRED, Wire, Wiretap, WiretapCentral, XSI

All other brand names, product names or trademarks belong to their respective holders.

Disclaimer

THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY AUTODESK, INC. "AS IS." AUTODESK, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

Contents

Overview	1
Toolpath point parameter functions	2
Sample macros	3
Using Toolpath Point Parameters mode	4
Segment and point identifiers	6
Postprocessor requirements	7

Overview

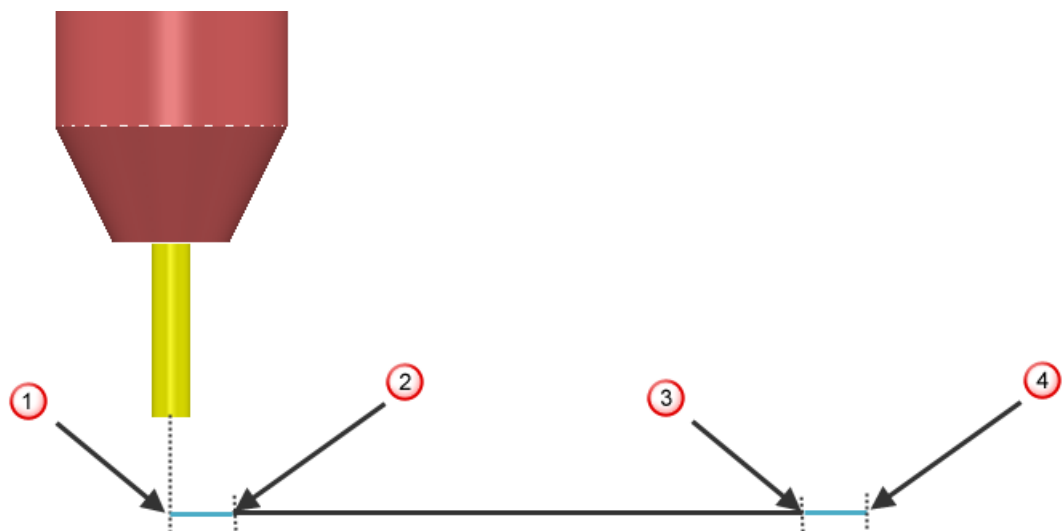
Adding parameters to toolpath points enables you to provide custom information to control different types of machines, such as:

- Laser cutters
- Water jet cutters
- Paint sprayers
- Additive machines



If you add parameters to toolpath points, you must update your postprocessor to process these parameters (see page 7).

For example, when using a laser cutter you may need to vary the intensity of the beam at certain points in the toolpath:



- ① — At the start, turn on the laser to a power of 90%.
- ② — 10mm from the start, turn the laser power to 100%.
- ③ — 10mm from the end, turn the laser power to 90%.
- ④ — At the end, turn the laser off.

Toolpath point parameter functions

You can use the following toolpath point parameter functions in a macro, or by typing them into a command window:

Description	Function
Returns the number of segments on the referenced toolpath.	<code>int toolpath_segment_count(entity ref)</code>
Returns the number of points on the n'th segment of the referenced toolpath.	<code>int segment_point_count(entity ref, int n)</code>
Returns the length of the n'th segment of the referenced toolpath.	<code>real segment_get_length(entity ref, int n)</code>
Returns the id of the point distance \pm tolerance from the start of the n'th segment on the referenced toolpath.	<code>int segment_get_point_at_distance(entity ref, int n, real distance, real tolerance)</code>
Creates a point at distance from the start of the n'th segment on the referenced toolpath. If there is an existing point within tolerance of this point then no new point is created. Returns the id of the newly created or existing point.	<code>int segment_create_point_at_distance(entity ref, int n, real distance, real tolerance)</code>

Returns a map of parameters on the m'th point on the n'th segment of the referenced toolpath.	<code>int point_parameters(entity ref, int n, int m)</code>
Adds a parameter named key with value value to the m'th point on the n'th segment of the referenced toolpath.	<code>int point_add_parameter(entity ref, int n, int m, string key, string value)</code>
Remove parameter named key from the m'th point on the n'th segment of the referenced toolpath.	<code>int point_remove_parameter(entity ref, int n, int m, string key)</code>
Remove all parameters from the m'th point on the n'th segment of the referenced toolpath.	<code>int point_remove_parameters(entity ref, int n, int m)</code>
Returns the geometric point on a toolpath which is closest to a given point in space.	<code>OBJECT closest_point_on_toolpath(ENTITY toolpath, POINT point)</code>

Sample macros

This example adds a parameter called **laser** to every toolpath point and the parameter's value increases by ten, from one point to the next.



Points can have more than one parameter, or they can have no parameters at all.

```
int segments = toolpath_segment_count("toolpath", "1")
int segment = 0
int error = 0

while $segment < $segments {
    int points = segment_point_count("toolpath", "1", $segment)
    int point = 0
    int power = 0

    while $point < $points {
        $error = point_add_parameter("toolpath", "1", $segment,
        $point, "laser", $power)
        $point = $point + 1
        $power = $power + 10
    }

    $segment = $segment + 1
}
```

Using Toolpath Point Parameters mode

The toolpath point parameter functionality is mainly macro-driven. However, you can enter the following command to start Toolpath Point Parameters mode to work with toolpath point parameters interactively:

EDIT TOOLPATH POINT_PARAMETERS

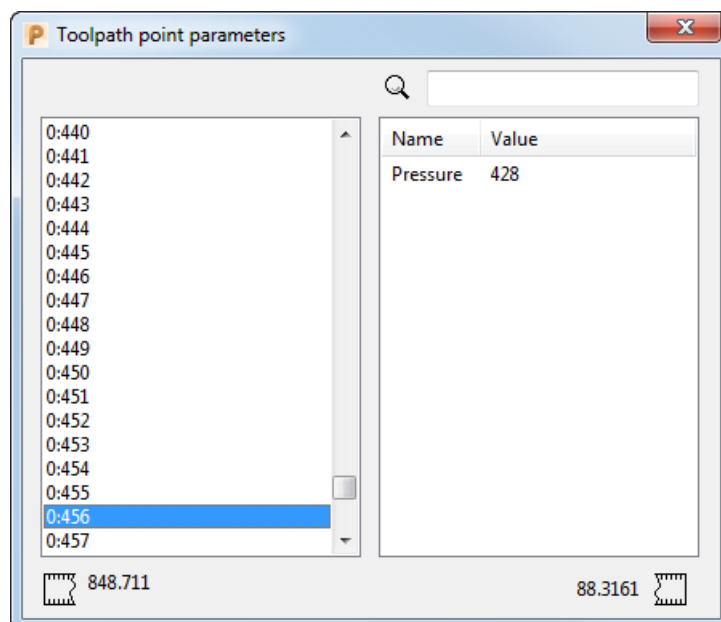


You must have an active toolpath to enter this mode.

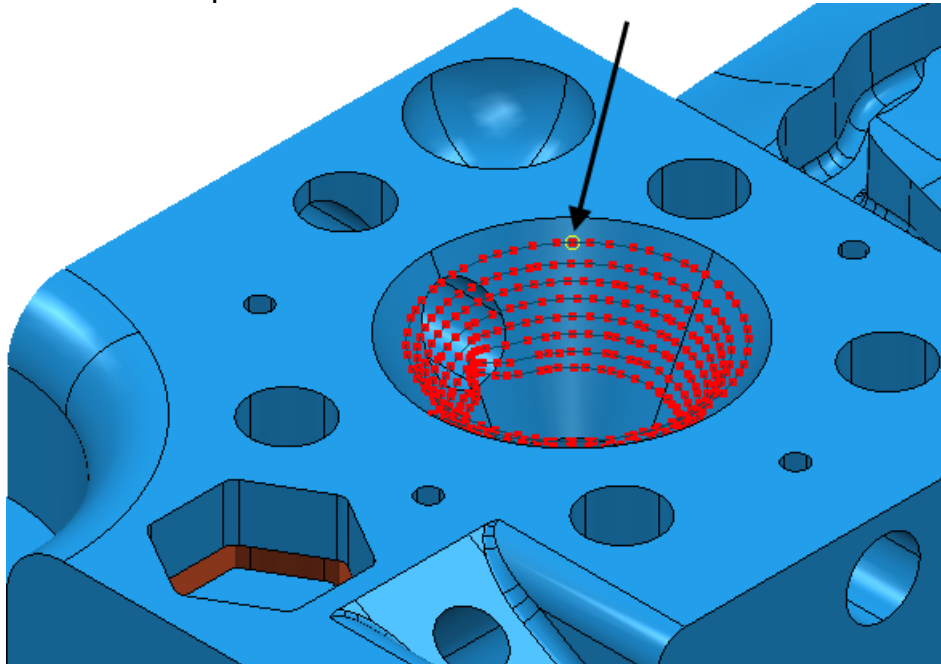
When in Toolpath Point Parameters mode, you can use the Toolpath Point Parameters toolbar:



The **Toolpath Point Parameters** dialog is also available:



The **Toolpath point parameters** dialog enables you to select a point and view its parameters.



Points with parameters are red and points with no parameters are blue. The selected point is surrounded by a yellow circle, in this case **0:456** where the first number (**0**) is the segment number and the second (**456**) the point number.



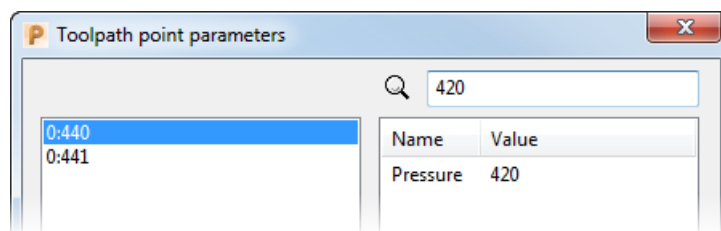
— Displays the distance of the selected point from the start of the toolpath segment.




— Displays the distance of the selected point from the end of the toolpath segment.



— Enter a value to filter the point list. The filter uses case-insensitive substring comparison. This helps you to find points with particular user data. For example, typing **420** in the filter box restricts the listed points to those where either the key, or value, contains 420.



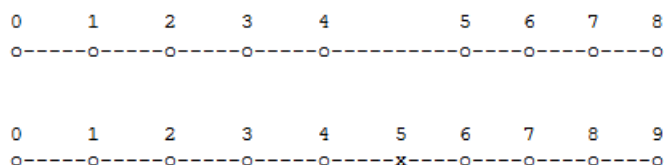
Closing the dialog, or clicking  in the mode toolbar, exits Toolpath Point Parameters mode.

Segment and point identifiers

When using toolpath point parameters, it is important to understand how segment and point identifiers work:

- Segment and point identifiers use zero-based indexing. This means the first point or segment has a value of **0**. If a toolpath has 10 segments, then the segment identifiers are 0 to 9 inclusive. Similarly, if a segment has 8 points then the point identifiers are in the range 0 to 7 inclusive.
- Segment and point identifiers are transitory, not fixed. So, if you create a new point, all the points after the additional point are renumbered.

In this example, a new point is inserted between points 4 and 5. In the updated toolpath segment, all points after point 4 have new identifiers. Because the number of points has also changed, a cached value from `segment_point_count()` is no longer correct.



Do not use any previously stored point identifiers or point counts after inserting a new point.

Postprocessor requirements

To use the parameters added to a toolpath point, you must update your postprocessor option file.

For example, if you create a parameter **laser**, you must modify your option file to include **udp_laser**.

If you need help with editing your option file, create a modification request using a **ppopt#** task.