PathDxf2GCode

Utility für grafische Definition von Fräspfaden

H.M.Müller, 11.12.2025

Inhaltsverzeichnis

1. Problem

Mit einem CAD-Programm konstruiert man zuerst einmal „Bauteile“, also die *Endergebnisse* eines Herstellungsprozesses. Als nächsten Schritt (oder parallel dazu) muss man auch diesen Herstellungsprozess selbst beschreiben und eventuell dafür weitere Konstruktionen zur Verfügung stellen – spezielle Werkzeuge, Zwischenschritte der Bauteile bei der Fertigung oder, im Falle einer CNC-Fertigung, „CNC-Programme“ zur Steuerung von CNC-Maschinen.

Bei CNC-Fertigung muss heutzutage typischerweise eine DXF-Datei in eine G-Code-Datei umgesetzt werden. Die Umsetzung hängt natürlich von der konkreten Fertigung ab, also u.a.,

- mit welcher Art von CNC-Maschine
- mit welchen Eigenschaften
- aus welchem Material
- mit welcher Genauigkeit

gefertigt wird: Für dasselbe Endergebnis (von der Form her) braucht ein 3D-Drucker eine andere G-Code-Datei als eine CNC-Fräse oder (wenn die Fertigung damit möglich ist) eine CNC-Drehbank. Aber nicht nur die Maschine ist relevant: Wenn eine CNC-Fräse ein Bauteil aus Blockmaterial herausfräsen soll, ist anderer G-Code nötig als bei einer Herstellung aus Halbzeugen. Und neben diesen grundsätzlichen Unterschieden muss ein G-Code-Generator auch viele spezifische Parameter der Ziel-CNC-Maschine wissen, um dafür korrekten und idealerweise auch effizienten G-Code zu erzeugen.

Die grundlegende Frage, die sich mir mit meinen speziellen Konstruktionen gestellt hat, ist: Kann ich ein Standardprogramm verwenden, oder brauche ich eine spezielle Lösung?

2. Herstellungsprozess

Um die Frage aus dem letzten Abschnitt zu beantworten, muss ich zuerst den angedachten Herstellungsprozess beschreiben, der wiederum durch das Endergebnis getrieben ist. Hier sind relevante Eigenschaften dieses Prozesses:

1. Was ich bauen will, sind einzelne, speziell projektierte mechanische *Modelle*, die aus vorkonstruierten *Bauteilen* zusammengebaut werden.
2. Aufgrund der relativ großen Anzahl verschiedener Bauteile ist deren Massenfertigung „auf Lager“ nicht sinnvoll – stattdessen werden bei der Projektierung die nötigen Bauteile eruiert und dann gefertigt.
3. Die Herstellung der Bauteile erfolgt aus vorher bekannten, einfachen Halbzeugen (aktuell sind das Alu-Flach- und Winkelprofile). Die G-Code-Erzeugung kann und soll auf diese Halbzeuge abgestimmt werden.
4. Die Bauteile sind in aller Regel klein (max. 100 mm lang, häufig unter 50 mm); die CNC-Fräse kann mit einem Aufspannen von Halbzeugen eine mittlere Anzahl (10...30) von Bauteilen fertigen. Es ist daher nötig, dass die *G-Code-Datei für einen Fertigungslauf* aus den Konstruktionsbeschreibungen mehrerer Bauteile *zusammengesetzt werden kann*.
5. Manche Bauteile müssen umgespannt werden (i.d.R. alle Winkelprofile, um die zwei Schenkelseiten getrennt zu bearbeiten).
6. In manchen Fällen ist ein Werkzeugwechsel nötig (von einem Schaftfräser zu einem Nutfräser).

3. Lösungskonzept

Vor allem die Halbzeuge, die ich auf meiner CNC-Fräse verwenden will, sind spezifisch für meine Konstruktionen. Es gibt sicher professionelle G-Code-Generatoren, denen man die Verwendung von Halbzeugen beibringen kann; die Lizenzkosten einerseits, und die Konfigurationsaufwände andererseits möchte ich lieber gar nicht wissen … (was vielleicht falsch ist, weil ich eine elegante Lösung übersehe; trotzdem riskiere ich das mal; hier könnte ich evtl. mal weiterforschen: How to Convert DXF to G-code: 4 Easy Ways | All3DP).

Darüberhinaus traue ich einem automatisch erzeugten G-Code nur begrenzt. Später einmal, wenn ich das Verhalten meiner Maschine wirklich verstehe, werde ich mich drauf einlassen – aber momentan will ich einmal jede Bewegung der Maschine „selbst vorgegeben haben".

Deshalb ist meine Entscheidung: Ich will den G-Code-Generator selbst schreiben.

Allerdings will ich nicht durch mehr oder weniger „Intelligenz" aus der endgültigen Beschreibung eines Bauteils die Fräspfade rückrechnen: Sondern auch diese Pfade von Hand konstruieren, mit demselben CAD-Tool, das ich für die Bauteile-Konstruktion verwende:

- Erstens weil das einfacher ist;
- aber auch wegen der erwähnten „Vorgabe"-Anforderung.

Damit sehen die Grobanforderungen für meinen Herstellungsprozess und dann den G-Code-Generator so aus:

a) Darstellung aller für die Fräspfade nötigen Pfadelemente im CAD-Programme; derzeit sind das:

- gerade Segmente,
- Kreissegmente und
- Bohrungen;

b) Darstellung aller für die Fräspfade nötigen Werte (z.B. Tiefe einer Bohrung, Höhe einer Leerfahrt) über Parameter grafischer Objekte, die

- einfach über das CAD-Programm angegeben werden können,
- sich dort optisch erkennbar unterscheiden; und
- stabil im Programm aus der erzeugten DXF-Datei ausgelesen werden können;

c) für jeden Pfad Ergänzung von Anfangs- und Endpunkten und einer auslesbaren Bezeichnung, die zur Verknüpfung von Pfaden in einer übergreifenden Zeichnung verwendet werden können.

Was ich *nicht* verlange, ist eine Anpassbarkeit an verschiedene Fräser-Durchmesser: Für einen anderen Fräser muss der Pfad neu konstruiert werden.

Die folgenden zwei Kapitel beschreiben

- den Vorlaufprozess, um die G-Code-Dateien zu erzeugen (Kapitel 4);
- und dann den internen Aufbau des G-Code-Generators (Kapitel 5).

4. Benutzerdokumentation

Anmerkung: Im Folgenden beschreiben Sätze mit „man“ Tätigkeiten des Konstrukteurs; während Sätze im Passiv Aktionen der Software beschreiben. Beispielsweise beschreibt „Man legt eine Datei an“ eine Tätigkeit, während „eine Datei wird angelegt“ eine Programmaktion meint.

4.1. Arbeitsprozess

Der Konstruktions- und Herstellungsvorgang mit PathDxf2GCode läuft so ab:

4.1.1. Bauteile konstruieren

Man konstruiert die Bauteile werden[1]. Die Layers[2] bezeichnet man dabei i.d.R. mit Nummern ohne Buchstaben am Ende, z.B. 2913 oder 2913.4.

Danach legt man getrennte Pfadzeichnungen für jede „Pfaddimension“ an. I.d.R. ist für Bauteile aus Flach-Halbzeugen nur eine Pfadzeichnung (ein Aufspannen) nötig, während für Bauteile aus Winkelprofilen zwei Pfadzeichnungen nötig sind (die ich üblicherweise mit L und R bezeichne). Auch für Bauteile, die mehrere verschiedene Fräser brauchen, sind mehrere Pfadzeichnungen nötig (z.B. Räder mit einer Rille, die mit einem T-Fräser erzeugt wird).

4.1.2. Projekt konstruieren

Für die mechanischen Modelle werde ich eine Gruppe von Bauteilen gemeinsam in einem Fräsdurchgang erzeugen. Daher ist dort eine übergreifende *Projekt-Konstruktion* nötig, die die Fräspfade für alle diese Bauteile miteinander verbindet.

Dazu legt man in einer neuen CAD-Datei je Pfaddimension Projekt-Pfadzeichnungen an, die die jeweiligen Pfade aller zu fertigenden Bauteile auf der Opferplatte (Aufspannplatte) anordnen und hintereinander verknüpfen. Die Projekt-Pfadzeichnungen zeigen auch die bemaßte Anordnung der Halbzeuge auf der Aufspannplatte sowie die Position des Fräskoordinaten-Prüfpunkts und -Ursprungs.

Wenn man nur ein Bauteil fertigen will, dann reicht die Bauteile-Pfaddatei als Input für die G-Code-Generierung.

4.1.3. Pfaddateien exportieren

Die Pfad-Zeichnungen exportiert man nun als DXF-Dateien. Die Verzeichnisstruktur kann beliebig sein (siehe dazu auch /d-Parameter auf S. 30), praktisch bewähren sich wohl zwei Strukturen:

- Für Projekte ohne vorkonstruierte Bauteile legt man jeweils ein neues Verzeichnis an, i.d.R. mit dem Datum im Verzeichnisnamen. Dort werden alle nötigen Pfaddateien abgelegt.
- Für vorkonstruierte Bauteile und daraus zusammengesetzte Projekte legt man
  - die Bauteil-Pfade in ein Bauteil-Verzeichnis (oder einige wenige),
  - die Projekt-Pfadzeichnungen wie oben in ein Projekt-Verzeichnis.

Für jede DXF-Datei erzeugt man auch eine entsprechende PDF-Datei und legt sie mit ab. Sie hilft

1 Ich verwende das (altertümliche?) Becker-CAD 2D; und konstruiere „klassisch“ über Risse, also nicht mit 3D-Objekten. Ist halt so.

2 In Becker-CAD heißen die Layers „Folien“.

später, die Halbzeuge abzulängen und aufzuspannen, Messfahrten für Z-Probes-Punkte durchzuführen und generell den Fräsprozess zu überwachen.

4.1.4. G-Code erzeugen

Man ruft nun PathDxf2GCode für die exportierten DXF-Dateien auf (siehe S. 30). Als Ergebnis entsteht jeweils eine G-Code-Datei mit demselben Namen, aber mit Erweiterung `.gcode`.

Wenn bei der Umsetzung Fehler auftreten, muss man die Pfad-Konstruktion im CAD-Programm korrigieren (siehe dazu S. 31) und erneut exportieren.

4.1.5. Fräsen

Für jede G-Code-Datei

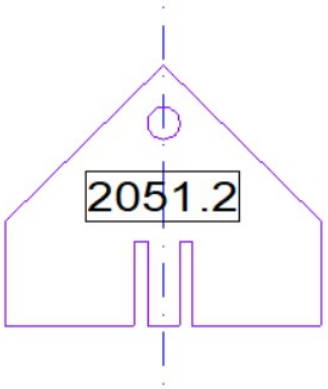
- spannt man nun die Halbzeuge entsprechend auf oder um(für diesen manuelle Arbeit dienen die PDF-Dateien mit den enthaltenen Material-Beschreibungen und Bemaßungen),
- setzt den der Koordinaten-Ursprung der Fräsmaschine
- und führt dann den Fräsdurchgang durch.

4.2. Pfadzeichnungen

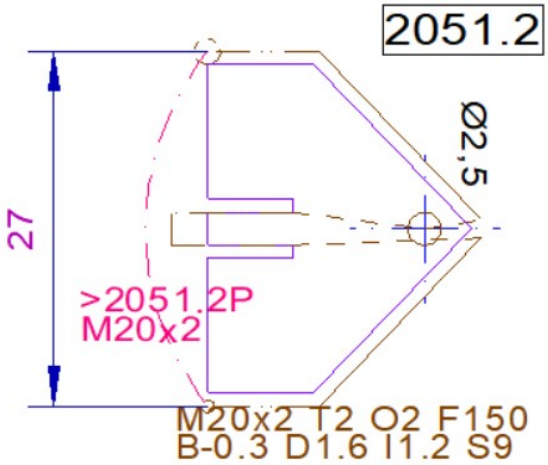
Pfadzeichnungen legt man einerseits für einzelne Bauteile an. Wenn – wie auf S. 7 beschrieben – diese Pfadzeichnungen zu Projekten zusammengesetzt werden sollen, legt man auch dafür Pfadzeichnungen an (die dann Teilpfad-Einbettungen für die Bauteile enthalten – siehe S. 20). Dieser Abschnitt beschreibt die allgemeinen Elemente von Pfadzeichnungen, ihre Verwendung steht in eigenen Abschnitten zu Bauteilen (S. 25) und Projekten (S. 27).

4.2.1. Ein Beispiel

Hier ist ein Beispiel einer Konstruktion – in diesem Fall eines Bauteils (Zeichnung Nr. 2050+2051 K; K steht für „Konstruktion“):



Die Zeichnung für den zugehörigen Fräspfad (hier ein *Bauteilpfad)* sieht so aus (Zeichnung Nr. 2051 P1; P steht für „Pfad“):



Einige wichtige Elemente jedes Pfades sind hier direkt erkennbar:

a) Der Anfang eines Pfades (unter dem Text M20x2...) ist durch einen kleinen Kreis mit 1mm Durchmesser markiert. Dort muss eine Reihe von Parametern angegeben werden, u.a. das Material (20x2-Flachprofil), die Oberkante des Halbzeugs (hier T2 = 2 mm über der Null-Ebene), der Fräserdurchmesser (O2=2mm) und weitere.

- Die grundlegende Idee ist, dass alle wesentlichen Parameter für das Fräsen an dieser einen Stelle beschrieben werden.

b) Für Kontur-Fräswege – insbesondere an geraden Kanten – wird der Bauteilpfad im Radius-Abstand des Fräsers (hier 1 mm, wegen des 2mm-Schaftfräsers) am Bauteil entlang gezeichnet.

c) Bohrungen werden mit ihrem echten Durchmesser im Pfad eingezeichnet (im Beispiel sieht man eine 2,5mm-Bohrung nahe der Bauteilspitze).

d) Das Ende des Pfades ist durch einen Kreis mit 2mm Durchmesser markiert.

e) Anfang- und Ende-Markierungen sind so angeordnet, dass der Bauteilpfad „gestapelt“ werden kann: Eine Kopie des Bauteilepfades – oder eines anderen Pfades für dasselbe Halbzeug – kann mit ihrem Anfangspunkt auf den vorherigen Endpunkt gelegt werden. Dadurch ist eine einfache Konstruktion der Projekt-Pfadzeichnungen möglich (siehe S. 27).

f) Eine strichpunktierte *Vertreterlinie* (_ . _) zwischen Anfang und Ende zeigt die Benennung des Pfades, die in eine Projektzeichnung übernommen wird. Diese Linie kann eine gerade Strecke oder ein Bogen sein; das ist nur eine optische Frage. Das >-Zeichen vor dem Pfadnamen wird im Abschnitt über Subpfade erklärt (siehe S. 20). An der Linie werden weitere Pfadparameter angegeben, die zur Überprüfung der Einbettung in Projekt-Pfadzeichnungen dienen (hier Materialangabe mit M).

g) Linienarten kennzeichnen die Art der Bewegung:

- ____ Durchgehende Linie = fräsen (Fräsfahrten)
- _ _ _ Strichlierte Linie = nicht fräsen (Leerfahrt)
- _.._.._ Strich-doppelpunktierte Linie = Halbfräsung; übliche Verwendungen sind:
  - An den Rändern, wo das Halbzeug sich fortsetzt, wird dadurch verhindert, dass dieses „abgefräst“ wird und damit die Aufspannung verliert. Im Beispiel weiter oben ist das oben und unten an den nicht-schrägen Segmenten zu sehen.

- An anderen Stellen dient es dazu, „Markierungen“ für Folgebearbeitungen in das Material zu fräsen – im Beispiel für die mit der Bandsäge anzubringenden schmalen Schlitze für das einzuschiebende Schlüsselrohr.
- _ _ . _ _ Doppelstrich-punktierte Linie = Stützsteg-Fräsung (siehe S. 19).

h) Das Pfaddiagramm erhält Bemaßungen der Außenmaße, um einerseits abzuschätzen, wie viel Material nötig ist; und um andererseits sicherzustellen, dass das Bauteil innerhalb des Arbeitsbereichs der Fräse platziert werden kann.

4.2.2. Pfadnamen

Jeder Pfad muss einen Pfadnamen haben, der dem regulären Pfadmuster-Ausdruck entspricht, der mit der Option /n an PathDxf2GCode übergeben wird. Der Standardwert dafür ist `([0-9]{4})[.]([0-9]+)([A-Z])`, gemäß diesem Muster muss ein Pfadname also aus drei Gruppen bestehen, deren erste zwei durch einen Punkt getrennt sind:

vierstellige Zahl . Zahl Großbuchstabe(nur A-Z)

Mögliche Pfadnamen gemäß diesem Standardmuster sind z.B. 1234.1A oder 2055.999B. Nicht erlaubt sind Pfadnamen wie 2345 (der Punkt und die darauffolgenden Gruppen fehlen), 3456.9 (der abschließende Großbuchstabe fehlt), 4567.8Ä (Umlaute sind als Buchstabe nicht möglich) oder 5678.9AB (ein Buchstabe zuviel am Ende).

Durch Angabe der Optionen `/n` und `/p` kann man auch ganz andere Pfadnamenmuster festlegen (siehe S. 30). Die Muster müssen aufeinander abgestimmt sein, weil über sie die passenden Dateien für eine Teilpfad-Einbettung gesucht wird (Details dazu siehe S. Fehler: Verweis nicht gefunden).

4.2.3. Pfadlayer

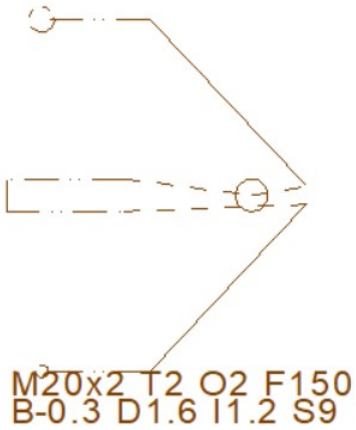
Was in der Pfad-Zeichnung nicht direkt zu sehen ist: Die Pfadlinien und Markierungen (aber nicht die Vertreterlinie für die Benennung – siehe S. 20) müssen auf einem gemeinsamen Layer[3] liegen, der mit dem Pfadnamen benannt ist, also hier 2051.2P.

Da auf diese Art in einer CAD-Datei ziemlich viele Layer entstehen (je Fräspfad einer, also i.d.R. je Bauteil einer oder zwei), müssen diese Pfadlayer sinnvoll organisiert werden. Folgender Struktur folge ich:

- Es gibt einen Haupt-Pfadlayer „Pfade“.
- Eine Pfad-Zeichnung für die Konstruktionen der Zeichnungen 1234 K erhält unter „Pfade“ einen *Gruppen-Pfadlayer* 1234.** (die zwei Sterne kennzeichen Gruppen-Pfadlayer, im Gegensatz zu Gruppen-Konstruktionslayern, die unter 1234.* gruppiert werden). Wenn eine Zeichnung mehrere Zeichnungsnummern umfasst, z.B. 1234 K und 2345 K, dann werden dafür getrennte Gruppen-Pfadlayer 1234.** und 2345.** angelegt.
- Unterhalb der Gruppen-Pfadlayer liegen die *Bauteil-Pfadlayer* für z.B. die Bauteile 1234.1, 1234.2 usw. Für Bauteile mit nur einem Pfadlayer wird dieser mit 1234.1P benannt, für Bauteile aus Winkelprofilen, die zwei Layer benötigen, wird 1234.2L und 1234.2R verwendet. In anderen Fällen (z.B. mehrere Pfade wegen Werkzeugwechsel) können auch andere Buchstaben verwendet werden.

Durch Anzeige nur des Bauteilpfadlayers kann man den Pfad optisch überprüfen. Für das Bauteil oben sieht das so aus:

3 In Becker-CAD: Folie.



Den Vertreterpfad (die strichpunktierte Linie) lege ich in den **-Pfadlayer des Bauteils (man könnte ihn in einen beliebigen Layer legen; beim Umkopieren in eine Projekt-Pfadzeichnung muss man ihn dort sowieso in den dortigen Projekt-Pfadlayer verlegen).

4.2.4. Aufbau einer Pfadkonstruktion

Nach dem beispielhaften Überblick in den letzten Abschnitten folgt hier die genaue Beschreibung, wie eine Pfadkonstruktion aufgebaut sein muss, damit sie von PathDxf2GCode verarbeitet werden kann.

Jede Pfadkonstruktion hat folgende Struktur:

Pfad mit Anfangs- und Ende-Markierung sowie mit Pfadelementen:

1. Leerfahrten
2. Fräsketten
    - Frässegmente: Strecken sowie Bögen; entweder volltief oder halbtief
3. Bohrlöcher
4. Helixkreise
5. Unterpfad-Einbettungen
6. Z-Probes

Pfade und ihre Elemente haben Parameter, die das Fräsen aussteuern. Z.B. haben Frässegmente eine Frästiefe, Fräsketten eine Zustellung, Helixkreise einen Durchmesser. Manche der Parameter werden direkt geometrisch angegeben (etwa der Lochdurchmesser), andere werden durch Texte bestimmt, die über dem jeweiligen Element liegen (siehe S. 12).

Manche Parameter können über die Pfadstruktur vererbt werden. So kann z.B. die Frästiefe für ein ganzes Bauteil beim Pfad bestimmt werden; wenn aber etwa einzelne Löcher als Sacklöcher gefräst werden sollen, kann dort die Frästiefe lokal verändert werden.

Die folgenden Abschnitte beschreiben die Pfadelemente und ihre Parameter sowie die Möglichkeiten ihrer Festlegung im Detail.

4.2.5. Parameter-Texte

4.2.5.1. Aufbau von Parameter-Texten

Ein Parameter-Text besteht aus einer Folge von Parameter-Definitionen, die selbst jeweils wieder aus einem Parameter-Buchstaben und einem Wert bestehen. Zwischen Buchstabe und Wert darf kein Leerzeichen stehen. Die einzelnen Definitionen werden durch Zeilenwechsel oder Leerzeichen getrennt. Wenn ein Wert sehr lang wird (das kann bei Variablendefinitionen leicht passieren; siehe S. 23), dann kann er mit einer Tilde (~) und eventuell folgenden Leerzeichen umgebrochen werden, z.B. hier bei der Definition der Werteliste für die Variable L:

```
O2 T2
S15 M...x2
:L5,10,15,~
   20,25
```

Der Ankerpunkt eines Textes muss momentan

- links unten
- rechts unten
- in der Mitte

liegen, und der Text darf nicht gedreht sein oder von unten nach oben oder von rechts nach links orientiert sein. Am besten wählt man den Ankerpunkt in der Mitte, weil sich dann bei Textskalierungen der Mittelpunkt des Überlappungskreis des Textes nicht verschiebt.

Die vorstehenden Bilder zeigen Definitionen von einer (N1), zwei (>8998.2P M20x0,1) und acht (siehe das Bild auf S. 11) Parametern. Die meisten Parameter sind Zahlenwerte; sie können mit Punkt oder mit Komma als Dezimaltrennzeichen geschrieben werden. Materialangaben können beliebige Zeichenfolgen sein, Pfadangaben nach > müssen dem regulären Ausdruck für Pfade entsprechen (siehe S. 27).

4.2.5.2. Zuordnung zu Objekten nur mit Überlappung

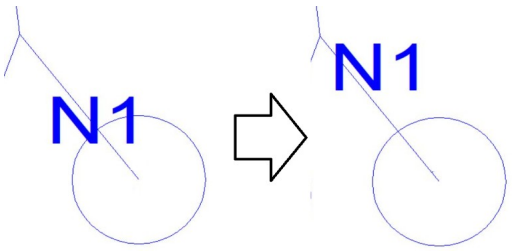
Parameter-Texte müssen so platziert werden, dass sie „ihr Element“ überlappen. Dabei muss darauf geachtet werden, dass

- der Text ebenfalls im Pfadlayer liegt;
- möglichst nur ein Text das Element überlappt. Allerdings versucht PathDxf2GCode bei mehrfachen Überlappungen, diese so aufzulösen: Ein Text wird möglichst dem nähesten Kreis zu geordnet; wenn es keinen überlappten Kreis gibt, dem nähesten Bogen; wenn auch kein solcher gefunden wird, der nähesten Strecke.
- die Überlappung weit genug ist: Die Überlappung wird durch einen Kreis um den Textmittelpunkt geprüft, der sich allerdings vor allem bei breiten Texten nicht über die volle Textbreite erstreckt.

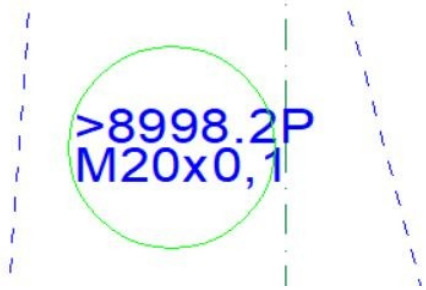
4.2.5.3. Problemfälle und ihre Lösung

Hier sind einige Beispiele für Problemfälle und ihre Lösungen:

1. Im folgenden Beispiel sollte sich der Text N1 auf die Strecke darunter beziehen; wegen der Bevorzugung von Kreisen funktioniert das aber nur, wenn man den Text ausschließlich über der Strecke anordnet:



2. Der (der hier von Hand ergänzte) „Suchkreis“ des folgenden breiten Texts ist kleiner als der Text: Obwohl das P die Linie überlappt, findet PathDxf2GCode diese Zuordnung nicht. Abhilfe schafft eine kleine Verschiebung des Textes nach rechts:

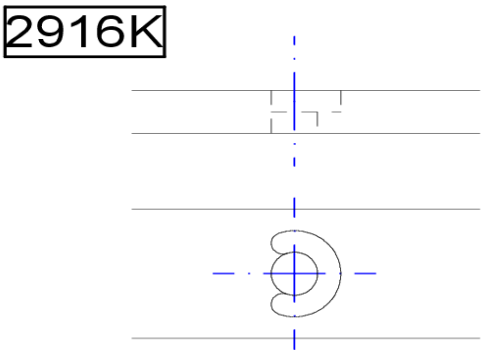


4.2.5.4. Zuordnung zu Objekten durch Parameter-Text-Layer

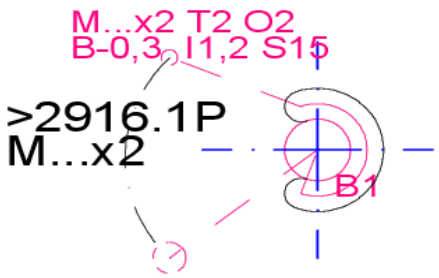
In manchen Fällen ist keine korrekte Zuordnung nur durch passende Überlappung möglich – etwa wenn sich ein Kreis zu nahe an dem Element befindet, das mit einem Parameter-Text versehen werden soll. Eine Lösung dafür bietet ein eigener *Parameter-Text-Layer*, in das sowohl das Element wie der Parameter-Text gelegt wird. Der Layername muss dazu mit einer Tilde und einem Buchstaben ergänzt werden:

```
Pfadlayer-Name~Kleinbuchstaben
```

Hier ist ein Beispiel dafür: Die folgende Konstruktion verbindet ein Durchgangsloch mit einem Stufe, die einen Teil des Lochs umfasst:



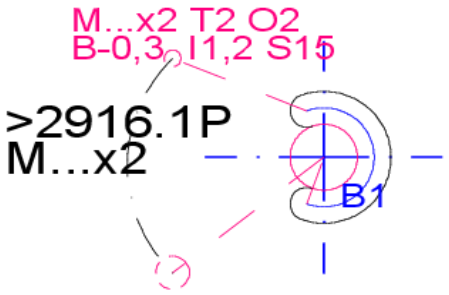
Als Fräspfad ist ein Helixloch sowie ein Bogen nötig, wobei der Bogen durch die Parameterangabe B1 nur bis zur Mitte des Profils gefräst wird. Alle Pfadsegemente und auch die Parameter-Texte liegen in der folgenden Zeichnung auf dem Pfadlayer `2916.1P`. der in einem roten Farbton dargestellt ist – auch der Bogen und der Text B1. Ohne spezielle Zuordnung wird der Text vom Kreis des Helixloches „gefangen“, statt dem Bogen zugeordnet zu werden:



Man kann das an der Ausgabe von PathDxf2GCode mit Option -x B1 sehen:

```
Objekt
   Circle LAYER=2916_1P CODENAME=CIRCLE TYPE=Circle LINETYPE=…
hat zugeordneten Text ''B1' @ [47.900|122.750] d=2.343'
```

Abhilfe erreicht man durch die Zuordnung von Bogen und Text B1 zu einem eigenen Paramstext-Layer `2916.1P~A` (der hier in der dafür üblichen blauen Farbe gezeigt wird). Dadurch kann der Text zuverlässig dem Bogen zugeordnet werden, obwohl er weitere Pfadsegmente überlappt:

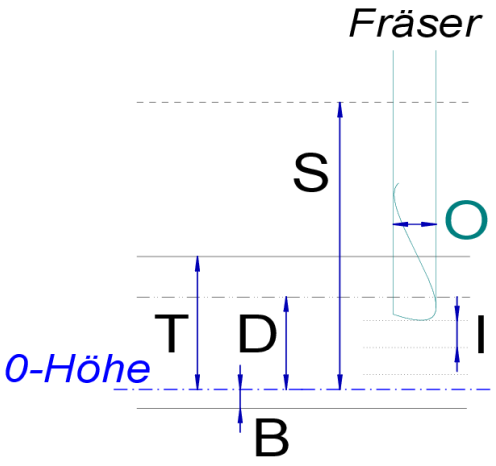


Die Ausgabe von PathDxf2GCode zeigt nun, dass der Text dem Bogen zugeordnet ist:

```
Objekt
   Arc LAYER=2916_2P~A CODENAME=ARC TYPE=Arc LINETYPE=CONTINUOUS%1 …
hat zugeordneten Text ''B1' @ [77.900|122.750] d=2.343'
```

4.2.6. Geometrie der Fräsparameter

Das folgende Diagramm zeigt die geometrischen Fräsparameter, die im Folgenden besprochen werden. Der B-Wert wird, wie alle anderen Werte, von der 0-Höhe nach oben gemessen; wenn wie hier gezeigt „unterfräst“ werden soll, muss B daher einen negativen Wert erhalten:



4.2.7. Pfade

Ein Pfad ist das grundlegende Element zur Beschreibung eines Fräsvorgangs. PathDxf2GCode kann nur zu einem ganzen Pfad eine G-Code-Datei erzeugen. Ein Pfad besteht aus einzelnen Pfad-Elementen, die graphisch die Fahrt eines Fräswerkzeugs beschreiben. Diese Elemente müssen daher direkt an einander anschließen, Lücken müssen durch Leerfahrten überbrückt werden. Als graphische Pfad-Elemente sind zulässig:≥

- Gerade Strecken (DXF: LINE)
- Bögen (DXF: ARC)
- Kreise (DXF: CIRCLE)

Außerdem werden Texte (DXF: TEXT und MTEXT) verwendet, um die Parameter der Elemente auszusteuern. Alle anderen Elemente in einer DXF-Datei werden ohne Fehlermeldung ignoriert. Wenn solche anderen Elemente (z.B. Splines) Teile von Pfaden sind, dann entsteht aus Sicht von PathDxf2GCode kein durchgehender Pfad, und man erhält daher i.d.R. die Fehlermeldung „*Keine weiteren Segmente ab Punkt ... gefunden*“ (siehe auch S. 31).

Der Anfang und das Ende eines Pfades müssen speziell gekennzeichnet werden. Dazu dienen kleine Kreise mit Linientyp Strich-Doppelstrich (DXF: PHANTOM):

- Der Anfang wird durch einen Kreis mit Durchmesser 1 mm markiert;
- das Ende durch einen Kreis mit Durchmesser 2 mm;
-

Am Pfadanfang (d.h. in einem Parameter-Text über dem 1mm-Anfangskreis) können folgende Werte festgelegt werden („TOMBIFDUPSRAW“):

- T[4]: Materialdicke in mm; Pflicht-Angabe, wenn nicht bei allen Segmenten einzeln angegeben.
- O: Fräserdurchmesser in mm; Pflicht-Angabe.
- M: Materialangabe; Pflicht-Angabe.
- F: Fräsgeschwindigkeit in mm/min; muss entweder beim Pfad oder als Aufrufparameter /f (siehe S. 30) angegeben werden.
- I: Zustellung in mm; muss entweder beim Pfad oder einzeln bei Fräsketten und Helixkreisen angegeben werden.
- B: Frästiefe (über 0-Ebene); bis zu dieser Tiefe werden Frässegmente, Helixkreise und Bohrlöcher gefräst, wenn dort nicht ein anderer B-Wert angegeben ist.
- D: Markierungstiefe; bis zu dieser Tiefe werden Markierungssegmente gefräst, wenn dort nicht ein anderer D-Wert angegeben ist.

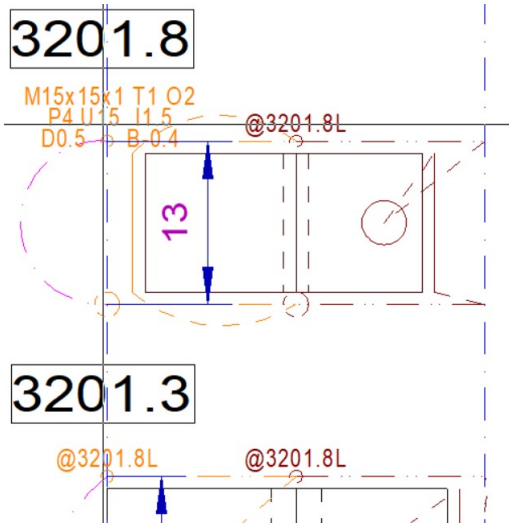
  Wenn sowohl B wie auch D angegeben sind, dann muss D größer als B sein. Das dient nur dazu, dass man D und B nicht „missbrauchen“ kann: B soll immer die „tiefe Fräsetiefe“ sein, D immer die „hohe Markierungstiefe“. Bei der Angabe von B oder D an einzelnen Segmenten (siehe S. 17) wird das aber nicht geprüft – dort kann man also mit B und D „tricksen“.
- P: Stützsteg-Länge; siehe Erklärung auf S. 19.

4 Die Kürzel stammen v.a. von englischen Bezeichungen ab: T = Top; O ist ein Durchmesser-Symbol; M = „Material“; F = „Feed“; I = „Infeed“; B = „Bottom“; D = „Depth“; S = „Sweep“; N = „Numbering“, P und U aus „Support“.

- U: Minimaler Stützsteg-Abstand; siehe Erklärung auf S. 19.
- S: Leerfahrt-Höhe; in dieser Höhe erfolgen Leerfahrten, wenn nicht bei einer Leerfahrt oder einem Frässegment ein abweichender S-Wert angegeben ist. Wenn die Angabe fehlt, wird
  - bei Subpfaden oder bei Angabe des /c-Aufrufparameters (siehe S. 30) die Summe aus T und O verwendet („ein kleines Stück höher als T“). Bei Verwendung dieses Defaults muss man sicherstellen, dass sich keine Befestigungsobjekte in der Arbeitsfläche des Pfades befinden;
  - beim Top-Level-Pfad (Projektpfad) der Wert des /s-Aufrufparameters (siehe S. 30) verwendet;
- R: Dateinamen-Ergänzung; dieser Text wird in die Namen der erzeugten Dateien aufgenommen. Der Zweck ist, über die Variablenbelegungen im Dateinamen zu informieren – siehe S. 24. Der Standardwert ist ein leerer Text, d.h. die Namen der erzeugten Dateien werden direkt aus dem Namen der DXF-Datei übernommen.
- Z: Fühlgeschwindigkeit in mm/min; bei fehlender Angabe wird der F-Wert verwendet (der allerdings in der Regel zu groß ist![5]).
- A: Maximaler Helixkreisdurchmesser in mm; bei fehlender Angabe wird als Wert 4 · O verwendet. Zur Verwendung dieses Wertes siehe S. 18.
- W: Erste Frästiefe für leerfahrtloses Fräsen (siehe S. 20); wenn diese Angabe beim Pfadstart steht, werden alle geraden und Bogen-Frässegmente leerfahrtlos gefräst (explizite Leerfahrten werden aber ausgeführt). Die nächsten Fräsfahrten werden wie üblich um die Zustellung I tiefer ausgeführt – ein “Bohren” auf diese nächsten Frästiefe muss also möglich sein, oder an einer schon ausgefrästen offenen Stelle durchgeführt werden.

4.2.8. Pfadtext als @-Referenz

Häufig haben Pfade in einer Zeichnung identische Parameter, z.B. für die Pfade auf der linken und rechten Seite eines Winkelprofils oder für Teile, die aus demselben Profil gefräst werden. Statt an den Anfangspunkten aller solchen Pfade immer alle Parameter anzugeben, kann man nur einen Pfad vollständig parametrisieren und an anderen Anfangspunkten auf diesen Pfad verweisen. Dazu gibt man dort nur *@Pfadname* an. Hier ist ein Beispiel:



5 Meine Erfahrung ist, dass bei F=150 mm/min die Abbremsung des Fräskopfs so langsam erfolgt, dass die Z-Probe-Messung um bis zu 0,3 mm abweicht. Bei F=50 mm/min erziele ich akzeptable Ergebnisse für Z-Probe-Messungen.

Eine solche @-Referenz darf keine weiteren Parameter enthalten, und der referenzierte Pfad muss sich in derselben DXF-Datei befinden.

4.2.9. Leerfahrten

Leerfahrten werden durch strichlierte (DXF: DASHED) oder lang-gestrichtelte (DXF: HIDDEN) Linien dargestellt. Für strichlierte Leerfahrten können folgende Parameter angegeben werden:

- S: Leerfahrt-Höhe; wenn nicht angegeben, wird der Wert beim Pfad verwendet.
- N: Reihenfolge bei Verzweigungen; siehe S. 25.

Lang-gestrichelte Leerfahrten akzeptieren keine Parameter-Texte. Sie sind in Situationen hilfreich, wo eine Pfadkonstruktionen viele Texte enthält, insbesondere für Projekt-Konstruktionen, die i.d.R. ausschließlich Teilpfad-Einbettungen (siehe S. 20 und 27) enthalten.

Leerfahrten können als gerade Linien oder Bögen dargestellt werden, aber auch bei der Darstellung als Bogen wird der Fräser in einer geraden Linie vom Anfangs- zum Endpunkt verfahren. Zur Beschleunigung werden mehrere Leerfahrten hintereinander, die mindestens in der Höhe S des Gesamtpfades verlaufen – von der angenommen wird, dass sie oberhalb aller Hindernisse liegt –, zu einer Leerfahrt zusammengefasst.

Die Geschwindigkeit einer Leerfahrt kann im G-Code nicht festgelegt werden, sie ist eine Eigenschaft der Fräse. PathDxf2GCode braucht diese Geschwindigkeit allerdings für die Statistikberechnung, dazu muss sie als /v-Aufrufparameter angegeben werden (siehe S. 30).

4.2.10. Fräsketten und Frässegmente

Frässegmente sind die hauptsächlichen „Arbeitstiere“ einer G-Code-Datei. Für jedes Frässegment müssen i.d.R. mehrere Fräsdurchgänge gemacht werden, bei denen der Fräskopf jeweils um eine kleine Distanz „zugestellt“ wird. PathDxf2GCode fasst *aufeinanderfolgende Frässegmente* zu *Fräsketten* zusammen und führt die Fräsdurchgänge jeweils für ganze Fräsketten durch. Frässegmente sind dabei nur gerade Strecken und Bögen, an Bohrlöchern, Helixkreisen und Leerfahrten endet eine Fräskette. Fräsketten können aus mehreren „Ästen“ bestehen („Stern-Ketten“), siehe dazu S. 25.

Die Fräsdurchgänge erfolgen in abwechselnder Vorwärts- und Rückwärtsfahrt. Fertige gefräste Segmente werden durch Leerfahrten ersetzen, die in der Leerfahrt-Höhe durchgeführt werden.

Die optionalen *Parameter einer Fräskette* werden beim ersten Frässegment angegeben; sie gelten dann für die ganze Kette:

- I: Zustellung in mm; wenn diese Angabe fehlt, wird der I-Wert des Pfades verwendet.
- W: Erste Frästiefe für leerfahrtloses Fräsen (siehe S. 20); wenn diese Angabe fehlt, aber beim Pfadstart vorhanden ist, wird die dortige Angabe verwendet.

Die Frässegmente können mit drei Linienarten gezeichnet werden:

- als durchgehende Linien (DXF: CONTINUOUS); für diese Fräsfahrten wird die Frästiefe mit dem B-Parameter festgelegt;
- als strich-doppelpunktierte Linien (DXF: DIVIDE) für „Markierungssegmente“; für diese Fräsfahrten wird die Frästiefe mit dem D-Parameter festgelegt;
- als doppelstrich-punktierte Linien (DXF: CENTER) für Frässegmente mit Stützstegen; für diese Fräsfahrten wird die Oberkante der Stützstege mit D, die Frästiefe mit B festgelegt. Zur Festlegung der Stützstege siehe S. 19.

Durch diese zwei Segmentarten können häufige Fälle verschieden tiefer Fräsungen ohne allzuviele Einzelangaben beschrieben werden, z.B.

- Fräsungen von Markierungen
- evtl. auch Fräsungen mit Haltestegen, wenn keine Markierungsfräsungen nötig sind.

Die *optionalen Parameter für einzelne Frässegmente* sind:

- F: Fräsgeschwindigkeit in mm/min; wenn nicht angegeben, wird die Angabe beim Pfad oder, wenn diese auch fehlt, des /f-Aufrufparameters verwendet.
- B: (bei durchgehender Linie) bzw. D (bei Markierungslinie): Frästiefe in mm; wenn nicht angegeben, wird die Angabe beim Pfad verwendet.
- N: Reihenfolge bei Sternen (siehe S. 25).
- Q: Kommentar, der in die G-Code-Datei übernommen wird.

4.2.11. Bohrlöcher

Bohrlöcher sind Löcher, die genau den Durchmesser des Fräsers haben (wo also der gezeichnete Durchmesser gleich dem O-Wert des Pfades ist); sie werden eher selten verwendet. Bohrlöcher können wie Frässegmente entweder mit durchgehender oder strich-doppeltpunktierter Linie gezeichnet werden (siehe S. 17). Für Bohrlöcher können folgende Parameter angegeben werden („FBQ“):

- F: Fräsgeschwindigkeit in mm/min; wenn nicht angegeben, wird die Angabe beim Pfad oder, wenn diese auch fehlt, des /f-Aufrufparameters verwendet.
- B oder D (je nach Linienart: Frästiefe in mm (Boden des Loches); wenn nicht angegeben, wird die Angabe beim Pfad verwendet.
- Q: Kommentar, der in die G-Code-Datei übernommen wird.
- (C – derzeit noch nicht unterstützt: Angabe, ab welcher Tiefe G81 statt G01 verwendet werden soll).

4.2.12. Helixkreise

Helixkreise sind kreisförmige Fräsungen mit einem größerem Außendurchmesser als dem des Fräsers. Der gezeichnete Kreis gibt den *Außendurchmesser* der Fräsung an; das ist hilfreich bei der häufigsten Anwendung, dem Fräsen eines Loches.

> Achtung: Alle anderen Fräspfade – insbesondere Bögen (ARC) – beschreiben die Bewegung des Fräser*zentrums*; nur bei den Helixpfaden wird der *äußere Rand der Fräsung* gezeichnet. Das ist verwirrend, weil man optisch einen (langen) Bogen nicht gut von einem Kreis unterscheiden kann. Ich lasse es aber wegen der häufigen Verwendung von Helixkreisen für Lochfräsungen so.
>
> Als eine Absicherung sind Helixkreise nur bis zu einem maximalen Durchmesser erlaubt, der 4 mal O (Fräserdurchmesser) beträgtr; für einen Fräserdurchmesser von 2 mm sind das also 8 mm. Wenn in Ausnahmefällen größere Helixkreise konstruiert werden sollen, dann kann man über den A-Pfadparameter den Wert anpassen.

Helixkreise werden durch eine spiralförmige Bewegung gefräst, die [derzeit] immer im Uhrzeigersinn (G02) erfolgt. Beim Ausfräsen eines Loches durch konzentrische Helixkreise von innen nach außen ergibt das ein Gegenlauffräsen.

Wenn ein Bauteil von außen gefräst werden und dabei Gleichlauffräsen vermieden werden soll, dann muss man den Fräspfad einzeln aus Bögen zusammensetzen, die in die gewünschte Richtung befahren werden.

Helixkreise können wie Frässegmente entweder mit durchgehender oder strich-doppeltpunktierter Linie gezeichnet werden (siehe S. 17). Folgende Parameter können angegeben werden („FBIQ/FDIQ“):

- F: Fräsgeschwindigkeit in mm/min; wenn nicht angegeben, wird die Angabe beim Pfad oder, wenn diese auch fehlt, des /f-Aufrufparameters verwendet.
- B oder D (je nach Linienart): Frästiefe in mm (Boden des Loches); wenn nicht angegeben, wird die Angabe beim Pfad verwendet.
- I: Höhe eines Spiralganges in mm; wenn diese Angabe fehlt, wird der I-Wert des Pfades verwendet.
- Q: Kommentar, der in die G-Code-Datei übernommen wird.

Für Helixkreise, deren Durchmesser größer ist als der doppelte Fräserdurchmesser, erzeugt PathDxf2GCode *keinen* G-Code zum Wegfräsen des entstehende Kerns – daher sind das auch *Kreise* und keine *Löcher*. Wenn das Wegfräsen des Kerns nötig ist (bei Sacklöchern oder zur Sicherheit gegen Verklemmen oder Wegschleudern eines ausgefrästen Kerns), dann müssen mehrere Helixkreise mit aufsteigendem Durchmesser konstruiert werden. PathDxf2GCode fräst solche konzentrischen Helixkreise auf jeden Fall von innen nach außen.

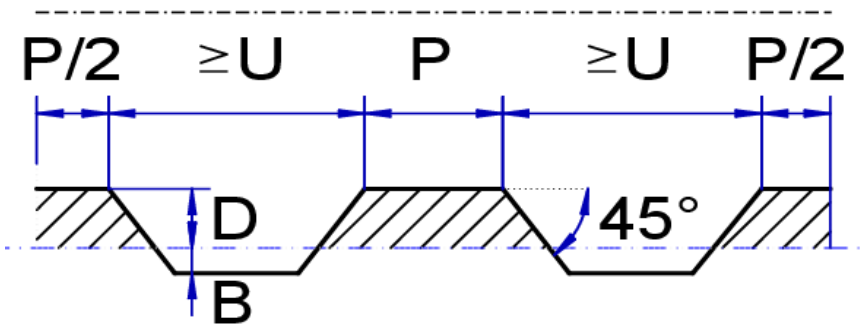
4.2.13. Stützsteg-Segmente

PathDxf2GCode unterstützt eine einfache Herstellung von Stützstegen: Eine doppelstrich-punktierte Linie (DXF: BORDER) führt zu einer kammartigen Bewegung des Fräsers, durch die bei passender Parameterwahl Stege des Materials stehenbleiben.

Stützstege sind möglich für

- gerade Linien
- Bögen
- Helixkreise.

Die Parameter und Geometrie der erzeugten Stege sind im folgenden Diagramm dargestellt:



Folgende Parameter müssen angegeben werden („PUBD“):

- P: Länge der Stützstege; die zwei äußersten Stützstege haben die halbe Länge (der Grund dafür ist, dass bei vollständigen Helixkreisen dadurch ein Stützsteg der Breite P entsteht).

- U: Minimaler Abstand der Stützstege. PathDxf2GCode versucht so viele Stützstege wie möglich in gleichen Abständen entlang der Linie unterzubringen; der Abstand zwischen zwei Stegen ist aber immer mindestens U.
- B: Frästiefe zwischen den Stützstegen
- D: Frästiefe der Stützstege.

Die Parameter werden üblicherweise beim Pfadstart angegeben, können aber bei jeder Stützsteg-Linie einzeln angegeben werden.

Die Stützstege werden für jede Linie einzeln berechnet. Es ist daher schwierig, für Fräsketten, die aus mehreren Frässegmenten zusammengesetzt sind, „schön angeordnete“ Stützstege zu erzeugen. In solchen Fällen ist es i.d.R. einfacher, wenn man die Stützstege mit Kombinationen aus Fräs- und Markierungssegmenten direkt konstruiert.

Die kammartige Stützsteg-Fräsung kann natürlich auch eingesetzt werden, wenn solche „Zahnstrukturen“ im Material gefräst werden sollen, also B größer als 0 ist.

4.2.14. Leerfahrtloses Fräsen

Leerfahrtloses Fräsen (für gerade Frässegmente und Bogen) ist nötig für besondere Fräser, z.B. T-Nut-Fräser. Solche Fräser können nicht „einfach so“ aus dem Werkstück gehoben werden, ihre Fräsfahrt muss daher genauer kontrolliert werden. Das wird durch die Angabe eines W-Parameters erreicht. Beim leerfahrtlosen Fräsen werden berechnete Rückwärtsfahrten während des Fräsens einer Fräskette (siehe S. 17) nicht durch Leerfahrten bewältigt, sondern durch Zurückfahren entlang der gerade gefrästen Segmente der Fräskette. Wenn der W-Parameter beim Pfad angegeben wird (siehe S. 15), dann werden alle Fräsketten leerfahrtlos gefräst.

4.2.15. Teilpfad-Einbettungen

4.2.15.1. Einbettung mit Fräsen

Für das Fräsen mehrerer vorher konstruierter Bauteile müssen deren Pfad-Zeichnungen in Projekt-Zeichnungen referenziert werden können. Dies erfolgt mit einer strichpunktierten Linie (DXF: DASHDOT) – sowohl eine gerade Strecke wie ein Bogen kann verwendet werden, relevant sind nur Anfangs- und Endpunkt.

Am Subpfad-Element können folgende Optionen angegeben werden („TOM“):

- >: Name des einzubettenden Teilpfades; Pflicht-Angabe. Der Teilpfad wird in allen passenden DXF-Dateien gesucht – siehe S. 27. Der Teilpfad wird vom Anfangs- zum Endpunkt eingebettet.
- T: Materialdicke in mm; Pflicht-Angabe.
- O: Fräserdurchmesser in mm; Pflicht-Angabe.
- M: Materialangabe; Pflicht-Angabe.

Für das Subpfad-Element gelten folgende Bedingungen:

- Die Angaben T, O und M beim Anfangskreis des eingebetteten Teilpfades müssen mit den entsprechenden Werten an der Einbettungsstelle übereinstimmen.
- Der Abstand von Anfangs- und Endpunkt des eingebetteten Teilpfades muss mit dem Abstand von Anfangs- und Endpunkt der Einbettungslinie übereinstimmen (d.h. eine Skalierung des eingebetteten Teilpfades ist nicht möglich).

- Die maximale Verschachtelungstiefe von Teilpfad-Einbettungen ist 9.

4.2.15.2. Einbettung ohne Fräsen für „Korrekturfräsungen“

Statt einer strichpunktierten Linie kann ein Teilpfad auch mit einer punktierten Linie (DXF: DOT) gezeichnet werden. In diesem Fall wird die Einbettung wie eine Leerfahrt ohne Parameter behandelt (siehe S.17).

Dies ist hilfreich, wenn für eine Projektdatei mit mehreren Bauteilen (siehe S.27) nur einzelne Bauteile neu gefräst werden sollen, etwa weil dort ein Konstruktionsfehler vorlag oder weil ein weiteres Bauteil in der Projektdatei ergänzt wurde: Man kann dann die Bauteilpfade der schon korrekt gefertigten Bauteile temporär punktiert zeichnen und erhält dann den G-Code nur für die neu zu fertigenden Bauteile; danach ersetzt man die Punktierung wird durch eine Strichpunktierung.

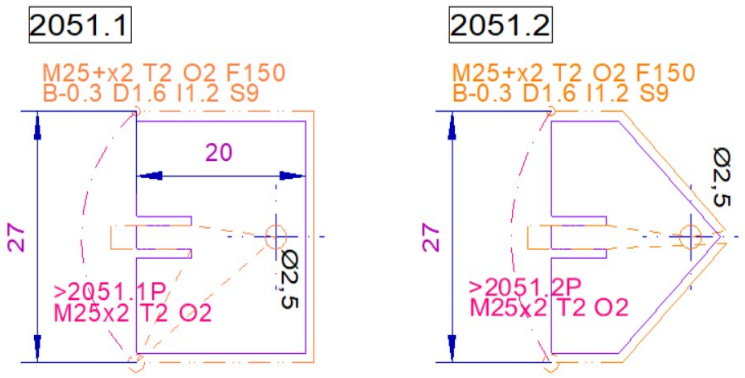
Auch in diesem Fall dürfen am Subpfad die Angaben für >, T, O und M angegeben werden, sodass für das „Überspringen“ der Einbettung alleine die Umstellung auf eine punktierte Linie reicht. Die Angaben werden aber vollständig ignoriert.

4.2.16. Teilpfad-Einbettungen mit Variablen

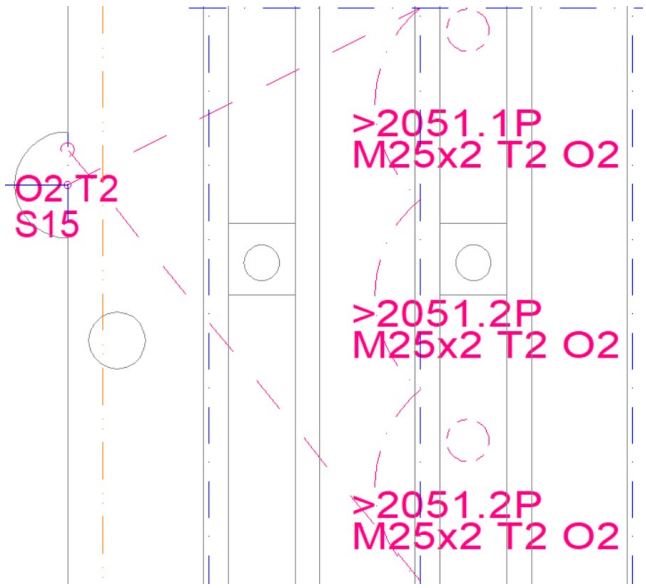
4.2.16.1. Beispiel für ein Aufspann- und Variablenprojekt

In vielen Fällen ist es hilfreich, wenn ein durch konstruiertes Projekt eine Vorlage ist, die bei konkreten Vorhaben parametrisiert werden kann.

Hier ist ein Beispiel: Neben dem spitzen Schlüsselgriff (siehe S. 8) mit Pfad 2051.2P haben wir auch noch einen rechteckigen Griff mit Pfad 2051.1P konstruiert:



Ein „Aufspannprojekt“ für einen rechteckigen und und zwei spitze Schlüsselgriffe könnte nun so aussehen:



Nun kann man sich fragen, ob man hier wirklich immer genau einmal 2051.1P und zweimal 2051.2P fräsen will – in einem Folgeauftrag braucht man vielleicht zwei rechteckige und einen spitzen Griff, oder drei spitze. Man könnte natürlich das ursprüngliche Projekt mit den 1+2 Griffen kopieren und in der Kopie die eingebetteten Pfade anpassen – aber lieber will man die Details des Aufspannprojekts nur einmal konstruieren; und dann die „Stückliste“ in einem eigenen Variableprojekt definieren, damit Änderungen am Aufspannprojekt nicht an mehreren Stellen nachgezogen werden müssen.

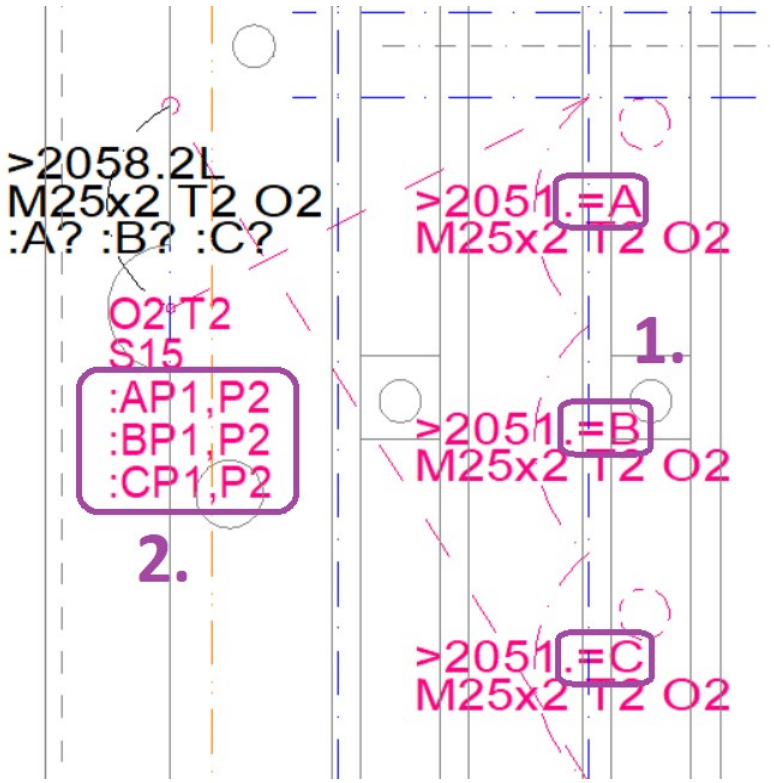
Die Lösung dafür ist, zwei getrennte Projekte zu haben:

1. Im Aufspannprojekt werden die Pfade der Griffe nicht mehr fest definiert, sondern als Variablen angegeben.
2. Das Aufspannprojekt definiert an seinem Startpunkt, welche Variablen und welche Werte zulässig sind.
3. In einem umgebenden Variablenprojekt werden den Variablen bestimmte Werte zugewiesen.
4. Die Namen der aus dem Variablenprojekt erzeugten Dateien enthalten eine Information über die verwendeten Variablenwerte.

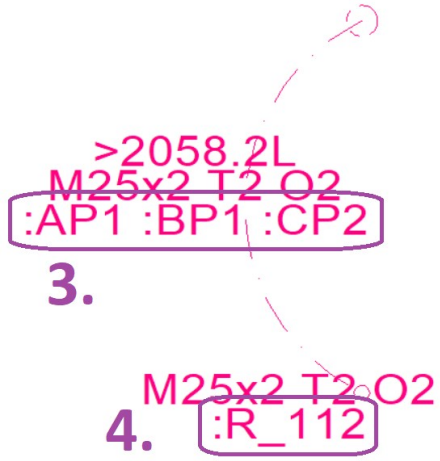
Hier sind diese Teile im Detail gezeigt:

Ad 1. Im Aufspannprojekt sind die eingebetteten Pfade über =-Variablen angegeben,

Ad 2. am Startpunkt werden die verwendeten Variablen und ihre möglichen Werte festgelegt. Die möglichen Werte für jede Variable sind hier P1 oder P2, was über komma-getrennte Listen angegeben wird:



Ad 3. Das Variablenprojekt besteht nur mehr aus einem Aufruf des Aufspannprojekts, wo man den Variablen Werte zuweist:



Ad 4. Mit Hilfe des R-Parameters ergänzt man im Pfad des Variablenprojekts eine (beliebige) Markierung, um über die gewählten Variablenwerte im Dateinamen zu inormieren. Hier ist das die Zeichenfolge _112, mit der man die gewählten Variablenwerte P1, P1, P2 symbolisiert.

4.2.16.2. Definition von Variablen

Variablen werden am Startpunkt eines Pfades durch Texte der Form

:<Name><Definition der zulässigen Werte>

definiert. Name ist ein einzelnes beliebiges Zeichen, üblicherweise sollen aber ein Buchstabe verwendet werden.

Zur Angabe der zulässigen Werte gibt es drei Möglichkeiten:

a) über eine komma-separierte Liste. Beispiele:

:AP1,P2,P3 bedeutet, dass für die Variable A die Werte P1, P2 und P3 zulässig sind.

:B2051.P1,2051.P2 bedeutet, dass für die Variable B die Werte 2051.P1 und 2052.P2 zulässig sind

b) über eine Liste von einzelnen Zeichen nach einem Fragezeichen. Beispiel:

:A?PQRS bedeutet, dass für die Variable A die Werte P, Q und R zulässig sind (was auch durch :AP,Q,R,S angegeben werden könnte).

c) über einen Bereich von Zahlen im Format von~bis. Beispiel:

:B-0,5~1,2 bedeutet, dass für die Variable Zahlen zwischen –0,5 und 1,2 (jeweils inklusive) zulässig sind.

4.2.16.3. Verwendung der Variablen

Innerhalb des Pfades, für den Variablen definiert sind, können diese in beliebigen Texten durch

=<Name>

z.B.

=A

eingesetzt werden. Wenn eine Variable verwendet wird, die nicht am Pfadanfang definiert ist, wird eine Fehlermeldung ausgegeben.

Die Werte der Parameter N und O können keine Variablen enthalten. Angaben wie N=X oder O=O, wo dann beim Subpfad-Element :X1 oder :O4 angegeben wird, sind daher nicht möglich und führen zu einer Fehlermeldung.

4.2.16.4. Setzen der Werte für Variablen

Bei der Teilpfad-Einbettung werden die Werte der Variablen gesetzt, die dann im Subpfad eingesetzt werden. Dazu werden am >-Text die Werte nach Doppelpunkt angegeben:

:<Name><Wert>

z.B.

:A1,5

Wenn ein Name angegeben wird, der beim Subpfad nicht definiert ist, oder ein Wert, der nicht mit der Definition der zulässigen Werte zusammenpasst, wird eine Fehlermeldung ausgegeben.

Der Wert einer Variable kann auch selbst wieder Variablen enthalten, die von weiter außen an den Pfad übergeben wird. Häufig wird das in der Form :B=B erfolgen, d.h. die nach „unten“ übergebene Variable B (linkes B) wird auf den von „oben“ erhaltenen Wert (=B) gesetzt.

Bei Verwendung von Variablenprojekten ist die Angabe des R-Parameters hilfreich: Sein Wert wird in die Namen der erzeugten Dateien übernommen, sodass man dort kennzeichnen kann, welche Variablenwerte zum erzeugten Gcode geführt haben.

4.2.16.5. Prüfwerte für Variablen

Wenn PathDxf2GCode mit der Option /c für Pfade mit Variablen aufgerufen wird, dann müssen

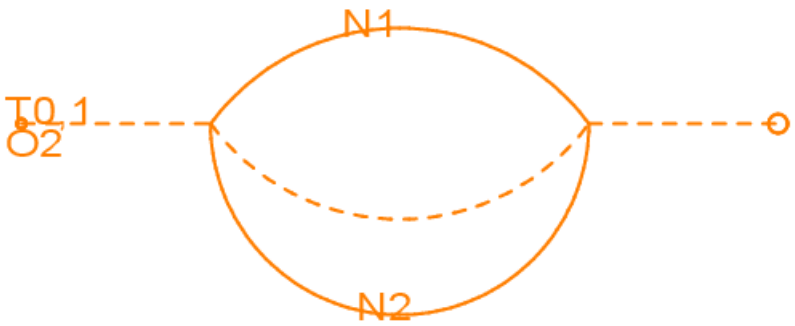
dafür ja Variablenwerte zur Verfügung gestellt werden, um korrekte Pfadtexte zu erzeugen. Auch dafür werden die Definitionen der zulässigen Werte verwendet:

- Wenn die zulässigen Werte über Komma- oder Fragezeichen-Listen festgelegt sind, werden der erste Wert der ersten Liste, der zweite der zweiten Liste, der dritte der dritten Listen usw. als Prüfwert verwendet. Wenn die Liste kürzer ist, wird in der Liste wieder von vorn begonnen. Zweck dieses Verfahrens ist, dass auch bei gleichartigen Liste (wie etwa im Aufspannprojekt oben) möglichst verschiedene Werte zur Prüfung verwendet werden.
- Für die Bereiche von zulässigen Werten werden abwechselnd von- und bis-Wert der Bereiche verwendet.

4.2.17. Sternförmige Pfade

Pfade dürfen auch „sternförmig“ sein, d.h., dass von einem Punkt aus in mehrere Richtungen weitergefahren (mit Fräs- oder Leerfahrt) werden soll. Damit in solchen Fällen ein eindeutiger Pfad berechnet werden kann, können die Segmente der Wege mit N-Texten versehen werden, z.B. N1, N2 usw. Segmente ohne N werden immer *nach* den mit N markierten Segmenten befahren, dabei werden kürzere Segmente nach vorne gereiht. Bohrlöcher und Helixkreise werden aber immer vor allen abgehenden Segmenten gefräst.

In manchen Fällen ist die Nummerierung mit N etwas trickreich, weil ein Segment an zwei Sternpunkten anschließt. Hier ist ein solches (als Testfall konstruiertes) Beispiel:



- Nach der Leerfahrt vom Startpunkt zum linken Sternpunkt wird das Segment mit N1 gefräst.
- Beim nun erreichten rechten Sternpunkt schließen zwei Leerfahrten ohne N an (die gerade zum Endpunkt und die gebogene nach links). Vorher werden aber die beiden mit N gekennzeichneten Fräsfahrten untersucht. Weil die obere davon schon gefräst worden ist und daher bei der Auswahl ignoriert wird, kommt daher das Segment mit N2 als nächstes in den Pfad.
- Diese Fräsfahrt erreicht nun wieder den Sternpunkt links. Da hier nun beide markierten Segmente „verbraucht“ (weil schon befahren) sind, wird nun die Leerfahrt in der Mitte gewählt.
- Am rechten Sternpunkt bleibt schließlich nur mehr die Leerfahrt zum Pfadende übrig.

4.3. Bauteilpfad-DXF-Dateien

Zum Einlesen durch PathDxf2GCode muss die CAD-Zeichnung als DXF-Datei abgespeichert werden. Da PathDxf2GCode später aus dem Namen eines Bauteilpfads in der Projekt-Pfadzeichnung die passende DXF-Datei herausfinden muss, muss jeder DXF-Dateiname über die enthaltenen Bauteilepfade Auskunft geben können.

Auf S. 10 ist beschrieben, dass dass ein Pfadname aus „Pfadnamen-Teilworten“ besteht; die genaue Zerlegung in Teilworte wird durch ein Muster (einen „regulären Ausdruck“) bestimmt. Wenn nun ein Pfad in Dateien gesucht wird, dann sucht PathDxf2GCode in den Dateinamen passende *Pfadnamenmuster*, die mit den Gruppen aus den Pfadnamen verglichen werden. Wenn die Muster in einem Dateinamen mit dem gesuchten Pfadnamen zusammenpassen, dann wird die Datei geöffnet und darin nach dem Pfad gesucht. Die Pfadnamenmuster müssen dafür einen bestimmten Aufbau haben, der durch die /p-Option festgelegt wird; ihr Standardwert ist `([0-9]{4})(?:[.]([0-9]+))?`, was folgende Pfadnamenmuster erlaubt:

- 4-stellige Zahl
- 4-stellige Zahl . Ziffern

also z.B. 1234 oder 1234.1 .

> Im Detail enthält das Muster zwei „Gruppen“, die durch einen Punkt getrennt sind; wobei die zweite Gruppe samt dem Punkt optional ist.

Ein DXF-Dateiname ist nun so aufgebaut:

> *...Pfadnummernbereich*{,*Pfadnummernbereich*...}....DXF

Ein Pfadnummernbereich hat dabei eine der folgenden zwei Formen und Bedeutungen:

- *Pfadnamenmuster*.
  - Das Muster zeigt im Dateinamen an, dass die Datei Pfade enthält, die mit den angegebenen Gruppen übereinstimmen. Ein Pfadnamenmuster 1234 mit nur einer Gruppe zeigt damit an, dass die Datei z.B. Pfade wie 1234.3A, 1234.12L usw. enthält, also Pfade, deren erste Gruppe gleich der angegebenen Gruppe 1234 ist. Ein Pfadnamenmuster 1234.5 im Dateinamen hingegen zeigt an, dass diese Datei Pfade wie 1234.5P oder 1234.5X enthält.
- *Pfadnamenuster–Pfadnnamenmuster*
  - Zwei Muster, getrennt durch ein Minus.
  - Ein solcher Bereich zeigt an, dass die DXF-Datei Pfade in diesem *Bereich* enthält. Dabei werden Gruppen als Zahl verglichen, wenn sie nur aus Ziffern bestehen (was beim Standardwert für /p immer der Fall sein muss), andernfalls werden sie als Text verglichen. So kann eine Datei mit Namen *1234–1236.DXF* beliebige Pfade enthalten, die mit 1234 oder 1235 oder 1236 beginnen; also u.a. den Pfad 1234.8A , aber auch den Pfad 1235.99B oder 1236.3C. Eine Datei mit Namen *1234.1–1234.10.DXF* kann u.a. den Pfad 1234.8A enthalten, aber auch den Pfad 1234.5B oder 1234.9A, aber nicht den Pfad 1234.40A.

Die Bereiche der Pfade von verschiedenen Dateien dürfen sich überlappen, z.B. dürfen zugleich die Dateien *1234.DXF* und *1234,1235.DXF* und *1230-1239,1241.DXF* vorhanden sein. Wenn PathDxf2GCode in diesem Fall z.B. einen Bauteilpfad 1234.2P sucht, wird es alle diese Dateien einlesen und durchforsten. Wenn dabei der Pfad allerdings mehrmals gefunden wird, gibt PathDxf2GCode eine Fehlermeldung aus.

Hilfreich ist es, wenn der Name einer Pfaddatei nicht nur aus Pfadnamenmustern besteht, sondern auch eine Bezeichnung enthält, z.B. *1234.1 Motorgehäuse.DXF*. Damit ist es auch leicht möglich, mehrere Dateien (Zeichnungen) für gleiche Nummernbereiche zu haben, z.B. eine weitere Datei *1234.1 Motorgehäuse-Befestigung.DXF*.

Zur Suche nach Subpfaden werden folgende Verzeichnisse durchsucht:

- Zuerst das Verzeichnis, wo die gerade verarbeitete DXF-Datei liegt;
- dann alle Verzeichnisse, die über /d-Parameter angegeben sind (siehe S. 27).

Der Dateiname soll in der Pfadzeichnung eingetragen werden, damit er beim DXF-Export ohne viel Nachdenken konsistent zu den Zeichnungsnummern gewählt wird.

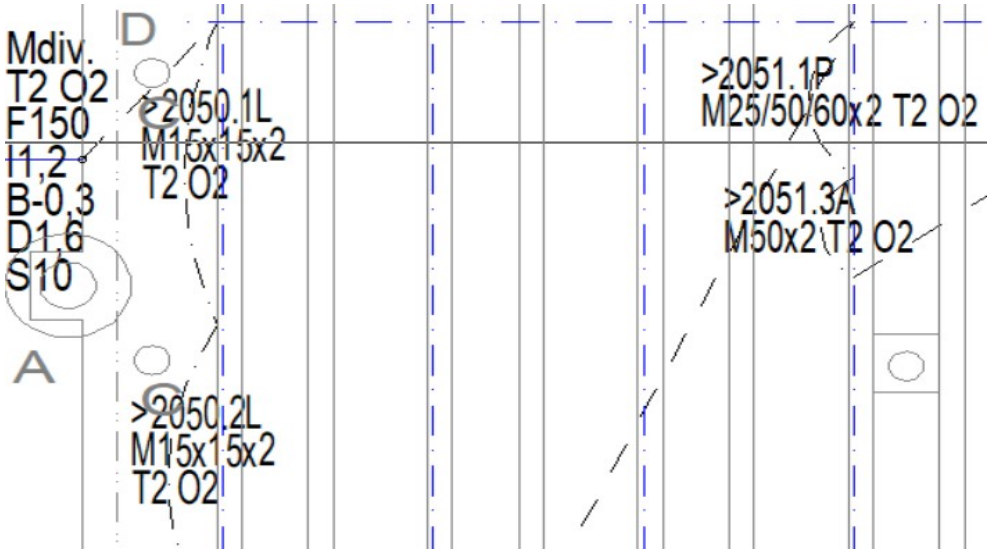
4.4. Projekte

4.4.1. Projektnummern

Wie erwähnt, werden für Projekte Pfad-Zeichnungen angelegt. Projekte erhalten Zeichnungsnummern von 8000 bis 8999. Die Projektnummern 8901...8999 sind für Testprojekte vorgesehen.

4.4.2. Projekt-Pfadzeichnungen und Teilpfade

Projekt-Pfadzeichnungen werden auf eine Kopie der Aufspann- und Opferplatte gezeichnet. Dies sieht dann z.B. so aus:



Bei einem Projektpfad müssen zumindest folgende Parameter angegeben werden:

- O, weil der Fräsdurchmesser für einen Projekt-Fräsdurchgang fest ist; und gegen die O-Angabe in eingebettetem Pfaden geprüft werden muss.
- T, weil aus Sicherheitsgründen immer klar sein muss, ab welcher Tiefe bei senkrechten Fahrten ein Bohren (G01) statt einer Leerfahrt (G00) nötig ist.
- üblicherweise S, weil Projektpfade praktisch immer Leerfahrten enthalten, deren Höhe definiert sein muss.

Die Konstruktion einer Projektpfad-Zeichnung erfolgt folgendermaßen:

a) Eine Kopie der Zeichnung 1084 Ph anlegen[6] und umbenennen. Die Namen der Projekt-Pfadzeichnungen haben folgende Form:

Projektnummer.Fräsdurchgang

6 In BeckerCAD kopiert man ein Blatt, indem man die MOD-Datei ein weiteres Mal öffnet und das zu kopierende Blatt mit „Hinzufügen“ übernimmt. Danach muss es umbenannt werden.

Die Fräsdurchgänge werden durchnummiert, nach dem Fräsdurchgang Buchstaben als Kennzeichnung der Aufspannung und/oder des Fräsers angehängt, z.B.

8001.1P und 8001.2P	für zwei Fräsdurchgänge verschiedener Bauteile
8002.1L und 8001.2R	für zwei Fräsdurchgänge derselben Bauteile mit Aufspannung L und R
8003.1LS, 8003.2LT, 8003.3RT	für drei Fräsdurchgänge derselben Bauteile mit Schaftfräser (S) und dann mit T-Nutfräser (T) für linke Aufspannung (L) und dann mit T-Nutfräser für rechte Aufspannung (R).

b) Die Teilpfad-Linien (siehe S. 20) samt ihnen zugeordneten Texten (insbesondere den Pfadnamen) der zu fräsenden Bauteile werden platziert. Dabei kann ein Anfang eines weiteren Bauteil-Pfades auf das Ende des vorherigen gelegt werden, wenn das vom Platz und vom Halbzeug her möglich ist.

- Die Vertreterlinien werden samt Text aus den Bauteile-Pfadzeichnungen kopiert[7], Anfangs- und Endemarkierungen (1- und 2mm-Kreise) dabei *nicht* übernehmen (Grund: es darf in einer Pfadzeichnung nur eine Anfangs- und eine Endemarkierung geben).

c) Unzusammenhängende Gruppen von Pfaden werden durch Leerfahrten (in der Regel langstrichlierte Linien) verbunden.

d) Anfangspunkt und Endpunkt (links außen) werden ebenfalls mit Leerfahrten angebunden.

e) Die Zeichnung wird als DXF-Datei exportiert; der Dateiname soll gleich dem Zeichnungsnamen sein (also etwa 8001.1P.DXF). Darüberhinaus sollen auch PDFs der Projekt-Pfadzeichnungen auf dem Steuerrechner der Fräse abgespeichert werden, da sie die Arbeitsunterlage für das Auf- und Umspannen der Halbzeuge und Bauteile und ggf. den Fräsertausch bilden.

4.5. Z-Probes

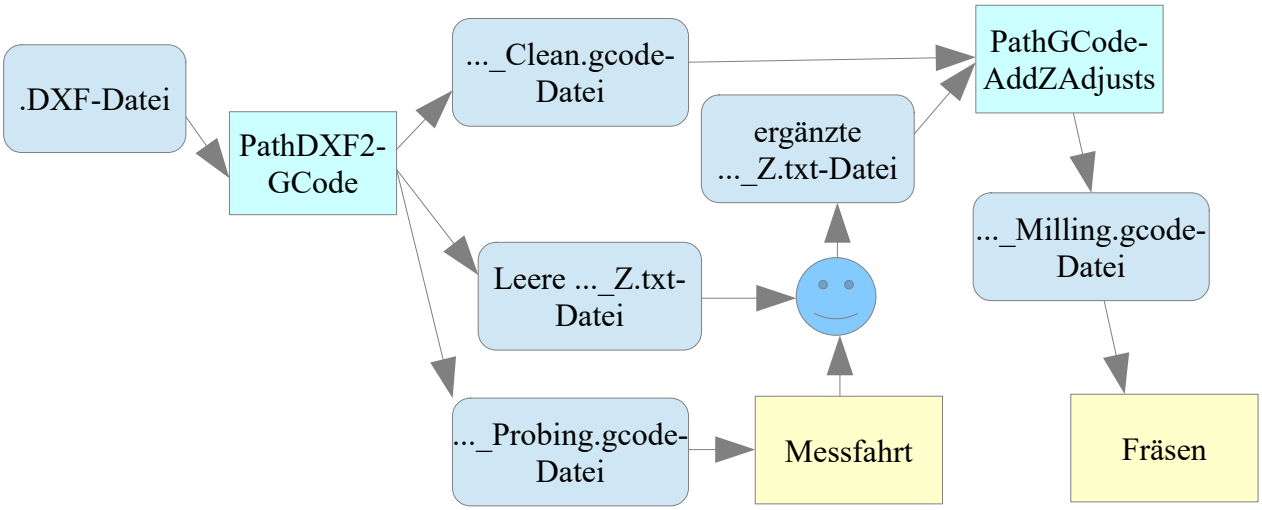
4.5.1. Grundlagen

An meiner Fräse habe ich ein Problem mit der Aufspannung: Auch auf der abgerichteten Aufspannplatte liegen die Halbzeuge nicht in einer Ebene, sondern sind bis zu etwa einem halben Millimeter tiefer oder höher. Bei voll durchgefrästen Profilen ist das egal, aber beim 3D-Fräsen nicht, und vor allem nicht, wenn ein Profil umgespannt und von der anderen Seite fertiggefräst wird. Dafür habe ich mir folgendes Hilfsmittel einfallen lassen:

- Eine Pfadzeichnung kann Z-Probe-Punkte enthalten, an denen vor dem eigentlichen Fräsvorgang durch eine Messfahrt der tatsächliche Z-Wert gemessen wird (der Fräser fährt die Punkte mit einem G38.2 an; den angezeigten Wert muss ich von UGS manuell in einer Textdatei erfassen).
- In der G-Code-Datei, die PathDxf2GCode erzeugt, wird bei jeder Z-Koordinate ein Formelausdruck hinterlegt, der die Korrektur des Z-Wertes in Abhängigkeiten von den Messwerten an den Z-Probe-Punkten beschreibt.
- Nach der Messfahrt wird mit Hilfe eines weiteren Programms namens PathGCodeAdjustZ die G-Code-Datei mit den Werten aus der Textdatei und den Formeln “nachberechnet”.
- Den Fräsvorgang führe ich dann mit dieser verbesserten G-Code-Datei durch.

7 In BeckerCAD am einfachsten so: Pfade und Texte in der Bauteil-Pfadzeichnung neben die Zeichnung kopieren; dann in der Projekt-Pfadzeichnung durch „Selektieren → Selektierte Elemente einfügen“ übernehmen; und dort entweder verschieben oder kopieren (wenn mehrfach nötig).

Der gesamte Arbeitsablauf sieht so aus:



4.5.2. Pfad-Zeichnung

Die Z-Probe-Punkte werden in der Zeichnung durch Kreise mit Linientyp Strich-Doppelstrich (“PHANTOM”) und Durchmesser 6 mm gezeichnet. Sie können an beliebigen Stellen liegen (also nicht unbedingt am Fräspfad). Den Pfad für die Messfahrt in der _Probing.gcode-Datei berechnet PathDxf2GCode selbst, indem vom Anfangspunkt immer der jeweils nächste, noch nicht besuchte Z-Probe-Punkt angefahren wird; zum Schluss wird zum Anfangspunkt zurückgekehrt.

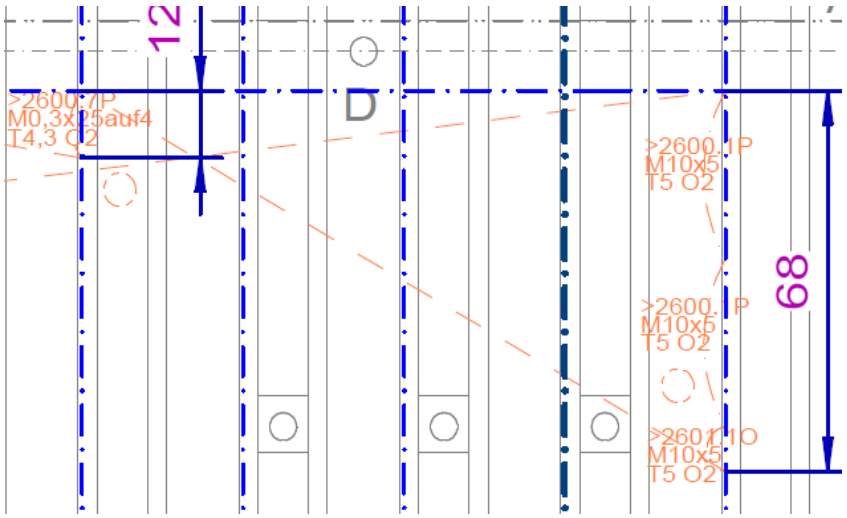
Die Z-Probes können auf zwei Arten angeordnet werden:

- außerhalb der Halbzeuge; dann muss bei der Messfahrt der Messfühler an die jeweilige Stelle gelegt werden.
- auf einem Halbzeug; dann muss bei der Messfahrt ein leitender Kontakt von Halbzeug und Messfühler hergestellt werden.

Bei einem Z-Probe-Punkt können folgende Parameter angegeben werden:

- T: Messfühler- oder Materialdicke in mm; wenn die Angabe fehlt, wird die entsprechende Pfadangabe am Anfangspunkt übernommen.
- L: Name des Z-Probe-Punktes, der in der _Z.txt-Datei angezeigt wird.
- Z: Fühlgeschwindigkeit in mm/min; bei fehlender Angabe wird die entsprechende Pfadangabe am Anfangspunkt übernommen, wenn auch die fehlt, wird der dortige F-Wert verwendet (der allerdings in der Regel zu groß ist).

Hier ist eine Pfadzeichnung, die zwei solche Z-Probes enthält (die orangen Kreise), jeweils in der Nähe von Subpfaden:



Die _Z.txt-Datei, die daraus entsteht, sieht so aus:

```
@[138.000|299.000] #2001=
@[242.000|264.000] #2002=
```

4.5.3. Messfahrt

Für die Messfahrt muss der Fräser mit dem einen Messanschluss verbunden werden. An den Z-Probe-Punkten muss, wie oben beschrieben, entweder der Messfühler platziert werden oder ein leitender Kontakt zwischen Halbzeug und Messfühler hergestellt werden.

Nach dem Start der _Probing.gcode-Datei fährt die Fräse alle Z-Probe-Punkte in der Reihenfolge an, wie sie in der _Z.txt-Datei vermerkt sind. Wenn der Fräser dabei den Messfühler oder das Halbzeug berührt, bleibt er jeweils 4 Sekunden stehen, damit man den Z-Wert in die _Z.txt-Datei übernehmen kann. Dazu wird dort manuell der in UGS abgelesene Z-Wert eingetragen, z.B. so:

```
@[138.000|299.000] #2001=5,1
@[242.000|264.000] #2002= 5.25
```

Als Dezimalzeichen sind hier sowohl Komma wie auch Punkt erlaubt; und vor und nach den Zahlen können gerne Leerzeichen oder Tabs stehen.

4.5.4. Erzeugen der Milling.gcode-Datei

Ein PathGCodeAdjustZ-Aufruf (siehe S. 32) erzeugt schlussendlich aus der _Clean- und der _Z.txt-Datei die endgültige _Milling.gcode-Datei.

4.6. Aufruf von PathDxf2GCode

4.6.1. Aufrufparameter

Beim Aufruf werden alle DXF-Dateien übergeben, für die G-Code-Dateien erzeugt werden sollen; zusätzlich müssen drei Pflichtwerte angegeben werden:

```
PathDxf2GCode /f150 /v1000 /s15 8001.27.1P.DXF 8001.27.2P.DXF
```

Die erzeugten G-Code-Dateien legt PathDxf2GCode im selben Verzeichnis wie die übergebenen DXF-Dateien ab.

Zusätzlich können folgende Optionen angegeben werden (siehe auch S. 15):

```
/h     Hilfe-Anzeige
/f 000 Fräsgeschwindigkeit in mm/min; Pflichtwert
/v 000 Maximalgeschwindigkeit für Leerfahrten in mm/min; Pflichtwert
/s 000 Vorgabe für Leerfahrt-Höhe in mm; Pflichtwert
/c     Überprüfen aller Pfade in der DXF-Datei ohne G-Code-Ausgabe; wenn /c nicht
       angegeben wird, dann darf die DXF-Datei nur einen Pfad enthalten
/x zzz Gibt für alle auf diese Regex passenden Texte aus, welchem DXF-Objekt
       sie zugeordnet sind
/d zzz Suchpfad für referenzierte DXF-Dateien
/n zzz Reg.Ausdruck für Pfadbezeichnungen in DXF-Dateien
       Standardwert ist ([0-9]{4})[.]([0-9]+)([A-Z])
/p zzz Reg.Ausdruck für Pfadbezeichnungen in Dateinamen
       Standardwert ist ([0-9]{4})(?:[.]([0-9]+))?
/l zzz Meldungssprache; Standardwert ist die Betriebssystemsprache
```

Beispielaufrufe:

```
PathDxf2GCode /h

PathDxf2GCode /f150 /v1000 /s15 /d..\Bauteile “2913 Ph.DXF“

PathDxf2GCode /f 150 /v 1000 /s 15 /c /d ..\Bauteile “2050-2051 P.1v.DXF“
```

4.6.2. Probleme und ihre Lösungen

4.6.2.1. „Pfaddefinition … nicht gefunden.“

Grund: Ein Pfad mit diesem Namen wurde nicht erzeugt.

Mögliche Auslöser:

- Anfangspunkt eines Pfades nicht mit Linienart „Strich-Doppelstrich“ (DXF: PHANTOM) versehen → Lösung: Anfangspunkt mit Linienart „Strich-Doppelstrich“ versehen.

4.6.2.2. „Ende-Markierung fehlt.“

Grund: Ein Pfad enthält keine gültige Ende-Markierung.

Mögliche Auslöser:

- Endpunkt eines Pfades nicht mit Linienart „Strich-Doppelstrich“ (DXF: PHANTOM) versehen → Lösung: Endpunkt mit Linienart „Strich-Doppelstrich“ versehen.

4.6.2.3. „S-Wert fehlt.“, „B-Wert fehlt.“, ...

Grund: Ein Segment kann den benötigten Wert nicht feststellen.

Mögliche Auslöser:

- Der Wert ist weder beim Pfadanfang noch beim Segment definiert → Lösung: Wert definieren.

4.6.2.4. „Keine weiteren Segmente ab Punkt … gefunden.“

Grund: Am angegebenen Punkt „geht es nicht weiter“.

Mögliche Auslöser:

- Ende ohne Fortsetzung (z.B. zwei Strecken treffen sich nicht in einem Punkt; oder eine Line

endet nicht genau im Mittelpunkt eines Loch-Kreises) → Lösung: Punkt in der Zeichnung finden[8] und Linien und Kreismittelpunkte exakt aneinander anschließen.

- Doppelte Elemente, die genau übereinanderliegen; nach dem Befahren eines Elements und der Rückkehr über das andere „geht es nicht mehr weiter“ → Lösung: Doppelte Elemente entfernen.
- Eventuell liegt eine Linie nicht im Pfadlayer → Lösung: Punkt in der Zeichnung finden und Linie in korrekten Layer verlegen.
- Im Pfad werden Elemente verwendet, die PathDxf2GCode nicht unterstützt (z.B. Splines) → Lösung: Punkt in der Zeichnung finden und ab dort Elemente durch unterstützte Elemente ersetzen (siehe S. 15).

4.7. Aufruf von PathGCodeAdjustZ

4.7.1. Aufrufparameter

Beim Aufruf werden i.d.R. die _Clean.gcode-Dateien angegeben werden, für die die Z-Korrektur durchgeführt werden soll. Die Namen der zugehörigen _T.txt-Dateien sowie der Ergebnisdatei (_Milling.gcode-Datei) werden daraus abgeleitet. Statt der _Clean.gcode-Dateien können auch die _Z.txt-Dateien und auch die DXF-Dateien, aus denen die _Clean.gcode-Dateien erzeugt wurden, angegeben werden:

```
PathGCodeAdjustZ 8001.27.1P_Clean.gcode 8001.27.2P.DXF
```

Zusätzlich können folgende Optionen angegeben werden:

```
/h     Hilfe-Anzeige
/m 000 Maximale Korrektur von Z-Werten; Pflichtangabe
/x zzz Regex für auszugebende Zeilen
/l zzz Sprache
```

4.7.2. Maximale Z-Korrektur

In der Praxis hat sich herausgestellt, dass ich die T-Werte von Z-Probes immer wieder falsch eingebe, z.B. T5 statt T2 beim Pfad. Wenn dann bei der Messfahrt etwa T=2,1 mm ermittelt und eingetragen wird, dann ergibt sich eine falsche Korrektur von 5–2,1 = 2,9 mm: Soviel taucht der Fräser tiefer ein – und prompt ist er abgebrochen. Um das zu verhindern, muss mit /m ein Maximalwert für die Korrektur angegeben werden. Typischerweise gebe ich hier 0,9 mm an.

4.7.3. Ausgabe der Statistikzeilen

Die Clean-Datei enthält gegen Ende hilfreiche Statistikzeilen mit der erwarteten Fräsdauer. Damit man diese nicht mit einem Editor heraussuchen muss, kann man an PathGCodeAdjustZ einen regulären Ausdruck übergeben, wonach auf diesen passende Zeilen ausgegeben werden. Die Statistikzeilen kann man mit `/x mm.*min` sichtbar machen.

8 Zum Finden der Problemstelle in BeckerCAD: „Punkt“ wählen (kleiner Kreis), dann werden bei Mausbewegungen die Koordinaten der aktuellen Position im Statusfenster mitgeführt.

4.7.4. Probleme und ihre Lösungen

4.7.4.1. “Zeile hat nicht das Format '(Kommentar) #...=Wert'”

Üblicherweise fehlen die Werte nach den Gleichheitszeichen.

Wenn man ohne Z-Adjustments fräsen will, dann kann man auch direkt die jeweilige _Clean.gcode-Datei an die Fräse senden.

4.8. Fräsen

Für den Fräsvorgang werden

- die erzeugten G-Code-Dateien
- und die zugehörigen PDFs der Pfad-Konstruktionen (sowohl Projekt-Pfadkonstruktionen wie Bauteil-Pfadkonstruktionen)

auf den Steuerrechner der Fräse übertragen.

Der folgende Fräsvorgang wird hier nicht mehr beschrieben.

5. Programmdokumentation

5.1. Überblick

PathDxf2GCode ist im wesentlichen ein in C# geschriebener 6-Pass-Compiler, der in mehreren Phasen den DXF-Input in den G-Code-Output transformiert. Alle 6 Phasen werden für jede Input-Datei vollständig durchlaufen; dabei werden eingelesene andere Dateien für eingebettete Pfade nur einmal gelesen und dann in einem Cache abgelegt.

Der eigene Code von PathDxf2GCode umfasst ca. 1500 NLOC, der von PathGCodeAdjustZ ca. 150 NLOC. Dazu kommt noch die (mit 19000 NLOC weitaus größere) netDxf-Bibliothek (siehe haplokuon/netDxf: .net dxf Reader-Writer (github.com)).

5.2. Github-Projekt

PathDxf2GCode ist auf GitHub unter https://github.com/hmmueller/PathDxf2GCode verfügbar.

5.3. Compiler-Phasen

Die 6 Phasen sind:

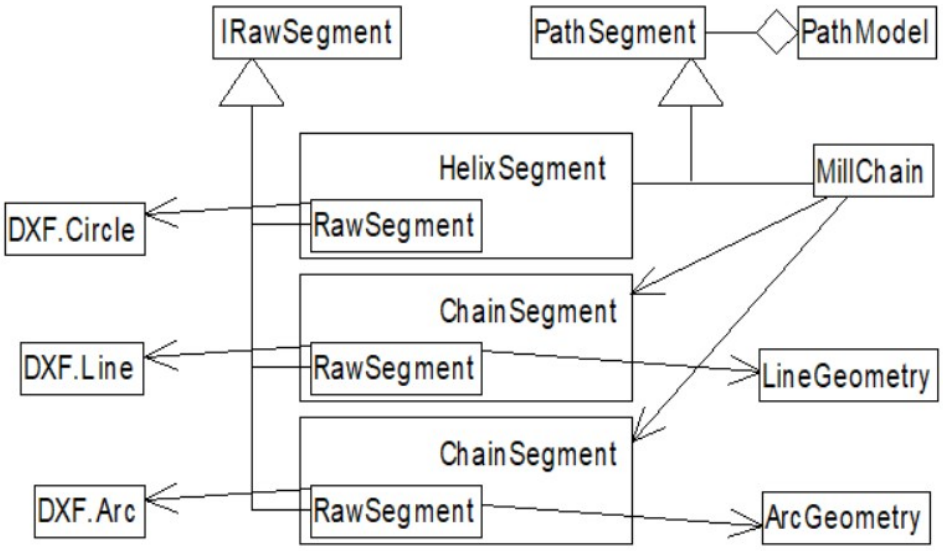
1. DXF-Datei → `DxfDocument`; dafür wird die netDxf-Bibliothek verwendet. Die zentrale Methode dieser Phase ist `DxfFile.LoadDxfDocument`.
2. `DxfDocument` → `RawPathModel`; das `RawPathModel` ist eine temporäre Darstellung der Pfade aus einer Pfadzeichnung. Jeder Pfad besteht aus
   - Pfadsegmenten (Typen: `MillSegment`, `SweepSegment`, `HelixSegment`, `DrillSegment` und `SubpathSegment`),
     - wo MillSegmente eine `MillingGeometry` vom Typ `LineGeometry` oder `ArcGeometry` haben;
   - Markierungen (Anfangs- und Endpunkt)
   - und Pfadparametern.

   Die zentrale Methode dieser Phase ist `PathModel.TransformDxf2PathModel`.
3. Zusammenfassung von Frässegmenten zu Fräsketten; die zentrale Methode dafür ist `PathModel.CreatePathModel`.
4. `PathModel` → `GCode`-Liste; die zentrale Methode dieser Textausgabe ist `PathModel.EmitGCode` zusammen mit den `EmitGCode`-Methoden in der `GCode`-Hierarchie.
5. Peephole-Optimizer der `GCode`-Liste (aktuell zur Zusammenfassung aufeinanderfolgender Leerfahrten); die zentrale Methode dafür ist `GCodeHelpers.Optimize`, zusammen mit der `GCode`-Klassenhierarchie.
6. Ausgabe der G-Code-Datei; zentrale Methoden sind in der Klasse `Program WriteGCodes` und `WriteZProbingGCode`, die `AsString`-Methoden der `GCode`-Hierarchie und die `WriteEmptyZ`-Methoden.

5.4. Grundlegende Datenstrukturen

Das folgende Diagramm zeigt beispielsartig ein Helix-, Strecken- und Bogensegment und die wichtigsten damit verknüpften Objekte:

netDxf-EntityObjects *RawSegments* *Segments*



5.5. Spezielle Datenstrukturen und Algorithmen in PathDxf2GCode

5.5.1. GCodeHelpers.cs

5.5.1.1. Optimize

Diese Methode ist ein Peephole-Optimizer für die erzeugte `GCode`-Liste. Die Erkennung von zu optimierenden Mustern erfolgt über reguläre Expressions, wofür jedes `GCode`-Objekt durch ein Zeichen (Propery `Letter`) repräsentiert wird.

- Aktuell ist ein Peephole-Optimizer für die Zusammenfassung von Leerfahrten (mit dazwischenliegenden Kommentaren) implementiert.

5.5.2. MillGeometry.cs

MillGeometryHelper.CreateSupportBarGeometries

Diese Methode erzeugt die Auf- und Ab-Verläufe für Stützstege (siehe S. 19).

5.5.3. Params.cs

Die ParamsText-Klasse ist „raw“, enthält also die Werte noch vor der Variableninterpolation und kann daher an die RawSegments angehängt werden. Die interpolierten Werten stehen in den Params-Klassen.

Params-Objekte haben einen Parent-Pointer:

- `ChainParams, SweepParams, HelixParams, DrillParams, SubpathParams, ZProbeParams` → `PathParams`

- `MillParams` → `ChainParams`

5.5.4. PathModel.cs

5.5.4.1. Innere Klasse RawPathModel

Rohe Modelle, die aus der DXF-Datei gelesen worden sind.

5.5.4.2. Innere Klasse Collection

Cache für RawPathModels und PathModels.

5.5.4.3. CollectSegments

Hier erfolgt

- das Aufsammeln aller EntityObjects auf Pfaden
- die Zuordnung von Parametertexten zu Objekten
- die Auswertung spezieller Marker (Start, Ende, ZProbes)
- das Laden von Subpfaden.

Daraus entsteht ein `RawPathModel`.

5.5.4.4. NearestOverlapping<T>, NearestOverlappingCircle, ….Line, ...Arc, CircleOverlapsLine, ...Arc, DistanceToArcCircle, GetOverlapSurrounding

Finden von Objekten für die Zuordnung von Parametertexten.

5.5.4.5. CreatePathModel

Hier wird das eigentliche `PathModel` aus dem `RawPathModel` aus den Ergebnissen der folgenden Schritte erzeugt:

A. Verkettung der RawSegmente zu einem Pfad

B. Erzeugen der PathSegmente

C. Aufbau von MillChains aus aufeinanderfolgenden ChainSegments (Mark- und MillSegments)

D. Erzeugen der Parameterobjekte

E. Verknüpfen der SubpathSegmente mit dem jeweils refernzierten Modell (das ggf. erzeugt wird)

5.5.4.6. CollectAndOrderAllZProbes

A. Aufsammeln aller ZProbes im Modell und eingebetteten Modellen.

B. Sortieren der ZProbes zu einer halbwegs kurzen „Rundreise“.

C. Benennen der ZProbes auf der Rundreise.

5.5.5. PathSegment.cs

5.5.5.1. Innere RawSegment-Klassen

RawSegments und PathSegments sind klar getrennt. Die ersteren sind nicht abhängig von Variablenwerten, die letzteren schon. Die RawSegments sind Factories für ihr jeweiliges PathSegment-Objekt. Die Klassen sind über Generic-Parameter eng miteinander und – wo nötig – mit ihren Params-Klassen „verknotet“.

5.5.5.2. MillChain.EmitGCode

Für jedes Segment einer MillChain werden Edges im I-Abstand erzeugt. Dann werden diese Edges möglichst knapp hintereinander abgefahren. Wenn eine Edge auf derselben Höhe anschließt, dann wird diese als nächste gefahren: Im Regelfall werden daher Edges einer Ebene abgefahren, bevor tiefere Schichten gefräst werden.

5.5.5.3. HelixSegment.EmitGCode

Eine Helix wird aus Halbkreisen zusammengesetzt, die jeweils um I/2 in die Tiefe fahren.

5.5.5.4. SubPathSegment.EmitGCode

Zwischen dem SubPathSegment und dem referenzierten Pfad wird eine `Transformation3` erzeugt, die die Koordinaten in das aufrufende Modell umrechnet.

5.5.6. Transformation2.cs

Der Winkel zwischen zwei Vektoren wird in netDxf nur über den Hauptast von arccos berechnet. Das ergibt immer nur Winkel-Werte zwischen 0 und π, was nicht korrekt ist. Mir ist nur eingefallen, das über einen „Drehversuch“ zu lösen: ersten Vektor um arccos und –arccos drehen und prüfen, welche Drehung tatsächlich den zweiten Vektor ergibt.

5.5.7. Transformation3.cs

Die Z-Koordinate wird nicht einer üblichen Transformation unterzogen, sondern durch ZProbes angepasst.

5.6. Spezielle Datenstrukturen und Algorithmen in PathGCodeAdjustZ

5.6.1. ExprEval.cs

Einfacher LL(1)-Parser für eine kleine Teilmenge der G-Code-Expressions, die für die Z-Adjustments verwendet wird. Als Klammerpaare sind sowohl […] wie auch (…) zulässig.

5.7. Aktuell bekannte oder vermutete Probleme

Keine, die mich besonders stören würden. Details siehe Github-Projekt.

5.8. Fehlende Features

Geplant ist die Verschiebung der T-Ebene bei Pfadeinbettungen.

6. Referenzen

6.1. Parameter-Buchstaben

Durchgestrichene Parameter werden noch nicht unterstützt.

	Bedeutung	**Einheit**	**Standardwert**
A	Größter Durchmesser für Kreise	mm	4 · O
B	Frästiefe	mm	
C			
D	Markierungstiefe	mm	
E			
F	Fräsgeschwindigkeit	mm/min	/f
G			
H			
I	Zustellung	mm	
J			
K			
L	Bezeichnung einer Z-Probe	Text	
M	Materialangabe	Text	
N	Reihenfolge-Angabe		
O	Fräserdurchmesser	mm	
P	Stützsteg-Länge	mm	
Q	Kommentar in G-Code-Datei	Text	
R	Dateinamen-Ergänzung	Text	keine
S	Leerfahrt-Höhe	mm	Projektpfad: /s Subpfade: T+O
T	Materialdicke	mm	
U	Minimaler Stützsteg-Abstand	mm	
V			
W	Frästiefe für leerfahrtloses Fräsen	mm	
X			
Y			
Z	Z-Probe-Geschwindigkeit	mm/min	F
>	Subpfad-Name	Text	
:	Variablendefinitionen und -wertezuweisungen	Text	
	Leerfahrt-Geschwindigkeit	mm/min	/s

6.2. Linienarten

Linienart	AUTOCAD-Name	Bedeutung	Parameter
----------	CONTINUOUS	Fräsweg	N I F B W
- . . - . . -	DIVIDE	Markierungsweg	N I F D W
- - . - - . - -	BORDER	Fräsweg mit Stützstegen	N I F B D P U W
- - - - - -	DASHED	Leerfahrt	N S
– – – – –	HIDDEN	Leerfahrt ohne Parameter	
- . - . - . - . -	DASHDOT	Subpfad	> M T O
…….	DOT	Subpfad als Leerfahrt	> M T O
– - - – - - –	PHANTOM	Spezialkreise: - 1 mm: Startpunkt - 2mm: Endepunkt - 6mm: Z-Probe-Punkt	T O M B I F D U P S A W T L Z