

Postprozessor in CNCezPRO anpassen

Schnellverfahren ohne Programmierkenntnisse

Ein Postprozessor ist ein Programm, welches geometrische Daten (z.B.: CAD-Dateien) mit zugehörigen technologischen Daten (z.B.: Schnittwerte, Werkzeugdaten) in ein CNC-Programm umwandelt oder umgekehrt, wie bei CNCezPRO.

Nachfolgend wird dargestellt, wie ein provisorischer Postprozessor ohne Programmierkenntnisse der Makrosprache erstellt werden kann.

Trotz Bemühung nach weitgehender Normung hat jede Steuerung ihren eigenen Dialekt und jede Maschine hat zusätzliche Funktionen aufgrund ihrer Bauweise, Ausbaustufen oder Zubehör.

So finden sich zwar die Standardbefehle G01, G02, M03, M04, usw. immer wieder, jedoch sind viele Befehle steuerungs- und maschinenspezifisch bedingt verschieden. Dazu bietet jeder Hersteller verschiedene Bearbeitungszyklen an, unter anderem, um sich von der Konkurrenz abzuheben, sogar die Standard-Bohrzyklen G81 bis G89 werden von den Herstellern unterschiedlich ausgelegt.

Wir beschränken uns hier auf Änderungen der Befehlslisten-Datei, wo lediglich die vom Referenz-Postprozessor abweichenden Befehle angepasst werden.

Dies ist relativ einfach zu bewerkstelligen, indem man die Befehlslisten vergleicht und lediglich die andersartigen Befehle ändert.

Natürlich muss hierbei berücksichtigt werden, dass auf diese Weise nicht alle Bearbeitungszyklen anpassbar und uneingeschränkt nutzbar sind.

Dazu wären profunde Programmierkenntnisse in der Makrosprache nötig.

In den nachfolgenden Beispielen werden nur einige oberflächliche Eingriffe in das Makro-Programm erklärt.

Im Ordner CNCezPRO\Machines ist zu sehen, dass pro Steuerung 5 Dateien vorhanden sind, nämlich mit den Dateiendungen TCM, TML, TRT, TSL und TTM.

Wir benötigen vorerst nur die Dateien mit den Endungen TML und TTM.

Beim Öffnen mit einem beliebigen Editor sieht man, dass die Datei mit der Endung TML eine Klartext-Datei ist, die in der Makrosprache geschrieben ist. Nach Änderungen muss diese zuerst kompiliert (in Maschinensprache übersetzt) werden, damit sie lauffähig ist.

Die Datei mit Endung TTM ist eine Auflistung der CNC-Befehle mit verschiedenen Parametern, die nach Änderung ebenfalls kompiliert werden muss.

CNCezPRO kompiliert diese Dateien bei der Nutzung des Maschinen-Assistenten und erzeugt automatisch die TCM-, TRT- und TSL-Dateien.

Eine nähere Beschreibung der Dateien ist in der CNCezPRO-Hilfe zu finden.

Vorgehensweise:

1. Kopieren der TML- und TTM-Datei des Referenz-Postprozessors (z. B.: SampleMill1) vom Ordner „Machines“ in einen anderen Ordner. Umbenennen der Dateien z. B.: in MeineMaschine.TML und MeineMaschine.TTM
Zurückkopieren nach „Machines“.
Mit dem Maschinenassistenten könnten die kompilierten Dateien auch direkt aus dem Referenz-Postprozessor erstellt werden, jedoch ist die obige Vorgehensweise sicherer, um die Originaldateien nicht zu verändern:
2. Die TTM-Datei (z. B.:MeineMaschine.TTM) mit einem beliebigen Editor öffnen, falls Sie eine Textverarbeitung besitzen, welche die Zeilen alphabetisch sortieren kann, nutzen Sie dies zur besseren Übersicht.
Die Liste ausdrucken.
3. CNC-Referenzliste in der CNCezPRO-Hilfe oder auf der Webseite von CNCezPRO bei „CNC“ mit der Befehlsliste Ihrer Steuerung vergleichen.
Änderungen in der ausgedruckten Liste eintragen.
4. Abweichende Befehle folgendermaßen ändern:
Beispiel: MeineMaschine fährt den Referenzpunkt mit G74 an, SampleMill1 fährt jedoch mit G28 (wie Fanuc-6M) zum Referenzpunkt.
G74 suchen und in der zweitletzten Spalte g74 in g28 ändern, dazu die anderen Parameter vom originalen G28 ebenfalls übernehmen, damit sich G74 tatsächlich wie G28 verhält.
Falls G74 nicht in der Liste ist, die Zeile einfügen.
Die geänderte (neue) Zeile sieht dann so aus: $G74^{8^{309^{5^0^{g28^0^}}$
Beschreibung der Parameter in der CNCezPRO-Hilfe.
Die Änderungen können direkt im Editor (Textverarbeitung) vorgenommen werden, beim Speichern dann darauf achten, dass Endung TTM stimmt.
Genauso gut können diese Änderungen auch im Maschinenassistenten eingegeben werden.
Eigentlich wird auf diese Weise der Postprozessor nur „betrogen“, indem man ihm mitteilt: „G74 sei G28“. Bei einem perfekten Postprozessor müssten die Funktionen in der TML-Datei geändert werden, was allerdings nur mit Kenntnissen in der Makrosprache möglich ist.
5. Maschinenassistent aufrufen, der u. a. die Befehlsliste zur Änderung anzeigt und nach verschiedenen Maschinenparametern fragt und zum Schluss mit dem eingebauten Compiler die TCM-, TRT- und TSL-Datei generiert.

Der neue (provisorische) Postprozessor ist erstellt und sollte auf alle geänderten Befehle getestet werden, bitte auch überprüfen, ob alle 5 Dateien im Ordner „Machines“ stehen.

Die Vorgehensweise wird hier an zwei Beispielmaschinen, Fräsmaschine Luxmill (Steuerung: Luxtronic) und Drehmaschine Traub (Steuerung:Traub-TX8D) beschrieben.

Dabei wird auch ein wenig auf die Makrosprache-Programmierung eingegangen.

Postprozessor für Luxmill anpassen

1. Auswahl des Referenz-Postprozessors nach Durchsicht der Funktionen in verschiedenen TML-Dateien. Hier wurde SampleMill1 (entspricht Fanuc 6M) ausgewählt, weil dieser den Funktionen der Luxmill am nächsten kommt.
Kopieren der TML- und TTM-Datei des Referenz-Postprozessors SampleMill1.TML und SampleMill1.TTM vom Ordner „Machines“ in einen anderen Ordner.
Umbenennen der Dateien in Luxmill.TML und Luxmill.TTM
Zurückkopieren nach „Machines“.
2. Beim Vergleich der Befehlslisten fällt folgendes auf:
Luxmill hat zusätzlich die Funktion G52 = Lokale Nullpunktverschiebung. Referenzpunkt anfahren ist bei Luxmill G74, bei SampleMill1 ist dies G28. Der Funktionsumfang dieser Maschine wurde später noch um einige Zusatzoptionen erweitert, nämlich: M20 pneumatischen Schraubstock öffnen, M21 Schraubstock schließen, M22 Blasluft ein (zum autom. Abblasen der Späne in der Maschine), M23 Blasluft aus, M26 Fertigmeldung an Extern (z. B.: an eine SPS-Steuerung, die daraufhin einen Roboter startet, der das fertige Werkstück entnimmt, wieder ein neues Rohteil einsetzt und danach wird die Fräsmaschine von der SPS wieder gestartet).
3. Wir ändern oder fügen die folgenden Zeilen der TTM-Datei hinzu:
G52^9^301^-1^0^g92^0^ SampleMill1 bietet nur G92 an, wir nehmen G92, weil es sich ähnlich verhält wie G52
G74^8^309^-5^0^g28^0^ wir setzen bei G74 die Werte von G28 (Referenzpunkt anfahren) ein
H^2^404^10^2^SETH^0^ H ist normalerweise nur für die Werkzeuglängenkorrektur-Nummer vorgesehen, deshalb war der ursprüngliche Typ-Parameter H^3^...für Integerzahl (Ganzzahl), wir ändern den Parameter in H^2^... (Dezimalzahl), da wir H ebenfalls für den seitlichen Fräserversatz in den Fräszyklen benötigen.
4. Diese Spezialfunktionen sollen der Vollständigkeit halber beispielhaft erwähnt werden, wenn Sie diese Optionen nicht haben, können Sie es weglassen
M20^9^400^-10^0^m11^0^
M21^9^400^-10^0^m10^0^ M20 und M21 steuern einfach ein Ventil an, welches einen pneumatischen Schraubstock öffnet oder schließt, bei SampleMill1 sind dies M10 und M11 (Klemmung Auf/Zu), deshalb nehmen wir hier die Werte von M10 und M11
M22^9^400^-10^0^m07^0^
M23^9^400^-10^0^m09^0^ M22 und M23 steuern ein Ventil an, welches eine Druckluftzufuhr öffnet oder schließt, bei SampleMill1 ist hierfür nichts vorgesehen. Wir nehmen hierfür die Werte von M07 und M09 der SampleMill1 für Kühlmittel 2 Ein/Aus
M26^9^400^-10^0^m00^0^, M26 steht am Ende eines CNC-Programms oder Programnteils. Die Maschine hält an und meldet an eine externe Steuerung, dass sie fertig ist. Weil dies einem Programmstopp am nächsten kommt, nehmen wir hierfür die Werte von M00.

5. Falls Sie es sich zutrauen: hier einige Änderungen in der Makrosprache, die auf C++ basiert.

Beim Ausprobieren der Verweilzeiten mit G04 stellen wir fest, dass die Steuerung SampleMill1 die Werte für X in Millisekunden anstatt in Sekunden abarbeitet.

Dazu gehen wir in die Datei Luxmill.TML und suchen G04, es erscheint folgender Programmteil (Funktion):

```

//*****
FUNCTION g04()
LOCAL
    temp

    clear_variables()

    gcode:=4

    p:=GETP()

    IF (p==UNDEFINED)
        THEN
            PROGRAMSTOP()
        ELSE
            //IF (x!=UNDEFINED)
            //    THEN
            //        temp:=1000*x
            //        DWELL(temp)
            //    ELSE
            //        temp:=p*1000 //p contains time in seconds
            //        DWELL(temp)
            //ENDIF
        ENDIF
    RETURN x
ENDFUNCTION
//*****
```

Wer Grundlagen in Englisch hat, kann auch ohne Kenntnisse der Makrosprache erraten, was in diesem Programmteil steht.

Hier sind die Programmzeilen so eingezogen, dass man die zusammengehörigen Anweisungen erkennt.

Dies verbessert vor allem die Übersicht bei Bedingungen („IF“, geht von IF bis ENDIF) und Schleifen („WHILE“, geht von WHILE bis ENDWHILE).

Hier ist zu sehen, dass eine Bedingung eine andere Bedingung enthält.

Die erste (äußere) Bedingung ist:

```

IF (p==UNDEFINED)
....
....
ENDIF
```

Die zweite innerhalb der ersten ist:

```

//IF (x!=UNDEFINED)
....
....
//ENDIF
```

Die zweite (innere Bedingung) ist mit den doppelten Schrägstrichen auskommentiert; d. h.: die Anweisungen sind zwar vorhanden, werden aber nicht ausgeführt.

Die Bedingungen enthalten Verhaltensweisen und Programmanweisungen.

Wir ändern (Änderungen in rot) die Funktion in folgender Weise:
 Luxmill benutzt auch X zur Bestimmung der Verweilzeit, deshalb wird
 $x:=GETX()$ hinzugefügt.
 Die erste (äußere) Bedingung wird um x erweitert, so dass es jetzt heißt:
 IF (p==UNDEFINED AND x==UNDEFINED)
 Teile der zweiten (inneren) Bedingung wurden auskommentiert, da sie für
 SampleMill1 nicht zutrafen.
 Die doppelten Schrägstriche wirken wie die Klammern im CNC-Programm, der
 nachfolgende Inhalt der Zeile wird nicht ausgeführt.
 Wir reaktivieren die innere Bedingung, indem wir die Schrägstriche löschen.
 Bei Luxmill wird die Verweilzeit X in Sekunden und P in Millisekunden
 angegeben, bei SampleMill1 wird P in Sekunden angegeben und wir ändern
 $temp:=p*1000$ in $temp:=p$

```

//*****
FUNCTION g04()
LOCAL
  temp

  clear_variables()

  gcode:=4

  p:=GETP()
  x:=GETX()

  IF (p==UNDEFINED AND x==UNDEFINED)
    THEN
      PROGRAMSTOP()
    ELSE
      IF (x!=UNDEFINED)
        THEN
          temp:=x*1000
          DWELL(temp)
        ELSE
          temp:=p //p contains time in milliseconds
          DWELL(temp)
        ENDIF
      ENDIF
    RETURN x
  ENDFUNCTION
//*****

```

Zum besseren Verständnis die Übersetzung eines Teils der Funktion G04,
 man kann so auch ohne Programmierkenntnisse die Funktion begreifen:

p:=GETP()	p sei der P-Wert aus der Funktion GETP (GETP heißt soviel wie: „hole P“)
x:=GETX()	x hinzugefügt, weil Luxmill auch X als Verweilzeit nimmt, sonst das selbe wie bei P
IF (p==UNDEFINED AND x==UNDEFINED)	<u>Äußere Bedingung:</u> wenn p und x nicht definiert sind
THEN	dann
PROGRAMSTOP()	Programmstopp
ELSE	andernfalls
IF (x!=UNDEFINED)	<u>Innere Bedingung:</u> wenn x ungleich undefiniert
THEN	(heißt soviel wie: wenn x definiert ist)
	dann
temp:=x*1000	sei temp x mal 1000
DWELL(temp)	verweile die Zeit temp
ELSE	andernfalls
temp:=p //p contains time in milliseconds	sei temp = p
DWELL(temp)	verweile die Zeit temp
ENDIF	<u>Ende der inneren Bedingung</u>
ENDIF	<u>Ende der äußeren Bedingung</u>

6. Da der Kreistaschenzyklus G75 der Luxmill häufig benötigt wird, kopieren wir den Zyklus G87 vom Postprozessor DIN-PAL-M.TML, fügen ihn in Luxmill.TML zwischen G74 und G76 ein und ändern ihn für unsere Zwecke ab. Wir ersparen uns hier einen Kurs in der Makrosprache, jedoch zum besseren Verständnis einige Erläuterungen zu G75.

Die Parameter:

X, Y = Startpunkt (Kreismittelpunkt)
Z = Endpunkt (Gesamttiefe)
H = seitlicher Fräserversatz in X und (oder) Y
D = Korrekturnummer des Werkzeugs
Q = Tiefenzustellung in Z-Richtung
F = Vorschub
R = Radius der fertigen Kreistasche

Beispiel:

N0150 G75 X50 Y70 Z-30 H15 D2 Q9 F400 R37

Bedingungen:

-Das Werkzeug muss vor Aufruf von G75 in Z-Richtung mit Sicherheitsabstand über dem Werkstück stehen, es darf auch schon in X und Y positioniert sein.

Ablauf:

-Das Werkzeug fährt im Eilgang in X und Y zum Kreismittelpunkt, wenn nicht schon vorher geschehen
-Das Werkzeug fährt im Kreismittelpunkt mit Vorschub um den Betrag Q auf die erste Z-Tiefe
-Das Werkzeug stellt seitlich mit Vorschub um den Betrag H in +X-Richtung zu und berücksichtigt dabei den Fräserradius D
-Der erste Kreis wird im Uhrzeigersinn gefräst
-Das Werkzeug stellt evtl. nochmals seitlich mit Vorschub um den Betrag H in +X-Richtung zu und berücksichtigt dabei den Fräserradius D
-Der zweite Kreis wird im Uhrzeigersinn gefräst
... und so weiter, bis der endgültige Radius R erreicht ist
-Das Werkzeug fährt mit Vorschub zum Mittelpunkt zurück
-Das Werkzeug fährt evtl. nochmals mit Vorschub um den Betrag Q auf die zweite Z-Tiefe, um dann wieder die Kreise zu fräsen
... und so weiter, bis die endgültige Tiefe Z erreicht ist
-Das Werkzeug fährt mit Vorschub zum Mittelpunkt zurück und dann mit Eilgang auf die Z-Position, wo es vor Aufruf von G75 stand.

Nachfolgend der Kreistaschenzyklus G75 und einige Beispiel-Übersetzungen
GETCURRENTX = hole den gegenwärtigen X-Wert (lese den X-Wert ein)
GETREGISTERVALUE = hole den Wert aus dem Werkzeugkorrekturspeicher
WHILE führt eine Schleife so lange aus, bis die Endbedingung erreicht ist
ENDWHILE = Ende der WHILE-Schleife
RAPIDABSTO = Fahre mit Eilgang im Absolutmaß nach... (Eilgang absolut nach)
LINEARABSTO = Fahre mit Vorschub im Absolutmaß nach... (Linear absolut nach)
ARCCWTO = Fahre einen Bogen im Uhrzeigersinn nach... (Bogen Uhrzeigers. nach)
ARC = Bogen, CW = clockwise (Uhrzeigersinn), CCW = counter clockwise (Gegenuhrzeigersinn)

Kreistaschenzyklus für Luxmill

```
//*****
```

```
//Circular Pocketing Circle  
FUNCTION g75()
```

```
LOCAL
```

```
    currx, curry, currz, cyclex, cycley,i,j,k,hr
```

```
    clear_variables()
```

```
    currx:=GETCURRENTX()
```

```
    curry:=GETCURRENTY()
```

```
    currz:=GETCURRENTZ()
```

```
    r:=GETR()
```

```
    d:=GETD()
```

```
    //apply tool radius compensation  
    r := r - GETREGISTERVALUE()
```

```
    h:=GETH()
```

```
    q:=GETQ()
```

```
    cyclez:=GETZ()
```

```
    cyclex:=currx
```

```
    cycley:=curry
```

```
        RAPIDABSTO(currx,curry,currz)
```

```
    z := currz - q
```

```
    WHILE ( z > cyclez ) DO
```

```
        hr := h
```

```
        WHILE ( hr < r ) DO
```

```
            //feed down to depth of cut
```

```
            LINEARABSTO(currx,curry,z)
```

```
            currx := cyclex + hr
```

```
            LINEARABSTO(currx,curry,z)
```

```
            //circular feed
```

```
            i:=0-hr
```

```
            j:=0
```

```
            k:=0
```

```
            ARCCWTO(currx,curry,z,i,j,k)
```

```
            hr := hr + h
```

```
        ENDWHILE
```

```
        //do final pass
```

```
        //feed down to depth of cut
```

```
        //LINEARABSTO(currx,curry,z)
```

```
        currx := cyclex + r
```

```
        LINEARABSTO(currx,curry,z)
```

```
        //circular feed
```

```
        i:=0-r
```

```
        j:=0
```

```
        k:=0
```

```
        ARCCWTO(currx,curry,z,i,j,k)
```

```
        //retract
```

```
        currx :=cyclex
```

```
        LINEARABSTO(currx,curry,z)
```

```
        //decrement z
```

```
        z := z - q
```

```
    ENDWHILE
```

```
    //do final pass
```

```
    hr := h
```

```
    WHILE ( hr < r ) DO
```

```
        //feed down to depth of cut
```

```
        //LINEARABSTO(currx,curry,cyclez)
```

```
        currx := cyclex + hr
```

```
        LINEARABSTO(currx,curry,cyclez)
```

```

        //circular feed
        i:=0-hr
        j:=0
        k:=0
        ARCCWTO(currx,curry,cyclez,i,j,k)

        hr := hr + h
    ENDWHILE
    //do final pass
    currx := cyclex
    //feed down to depth of cut
    currx := cyclex + r
    LINEARABSTO(currx,curry,cyclez)
    //circular feed
    i:=0-r
    j:=0
    k:=0
    ARCCWTO(currx,curry,cyclez,i,j,k)

    //retract
    currx := cyclex
    curry := cycley

    LINEARABSTO(currx,curry,cyclez)
    RAPIDABSTO(currx,curry,currz)

    SETX(currx)
    SETY(curry)

    clear_variables()

    RETURN x
ENDFUNCTION
//*****

```

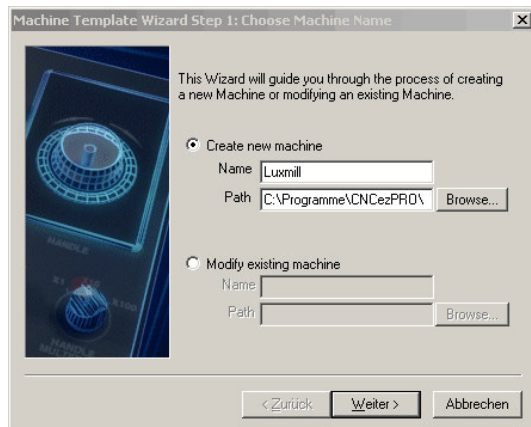
Alle diese Änderungen sind schon in den beigefügten Postprozessor-Dateien enthalten.

Für Vollständigkeit und Funktion kann keine Garantie übernommen werden.

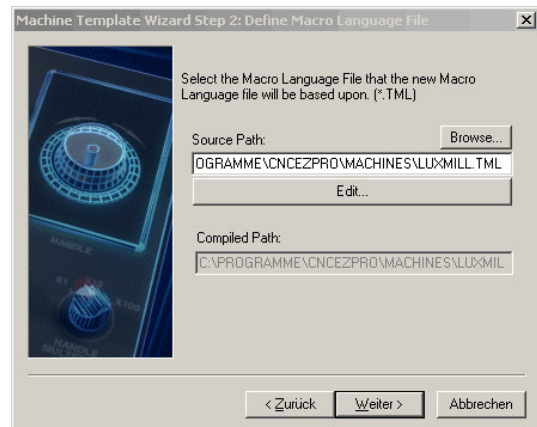
Der Autor haftet nicht für eventuelle Schäden.

Maschinen-Assistent benutzen und kompilieren - Luxmill

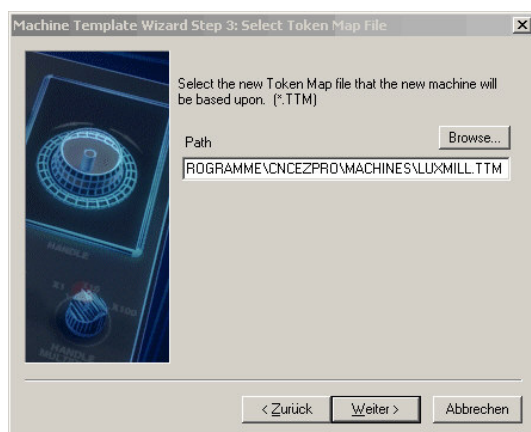
Im Menu Optionen > Maschinenassistent aufrufen



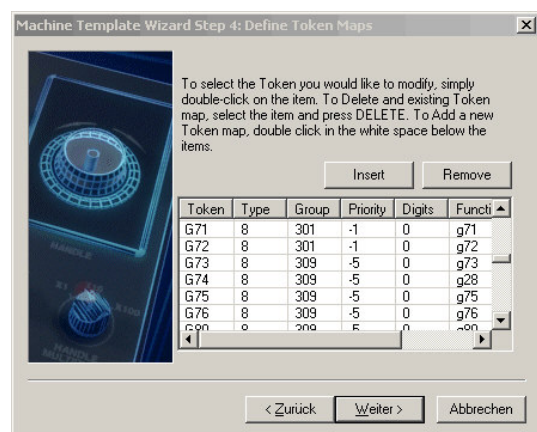
Schritt 1: Name der Maschine oder Steuerung eingeben.
Pfad wird automatisch vorgegeben, kann bei Bedarf geändert werden.



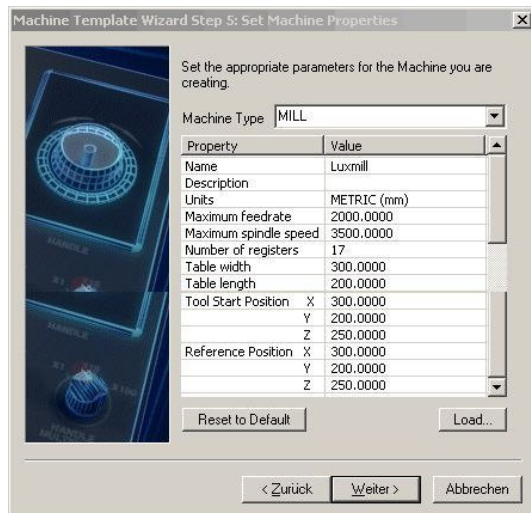
Schritt 2: Als Vorlage die zuvor kopierte TML-Datei suchen.
Hier: Luxmill.TML im Ordner „Machines“.



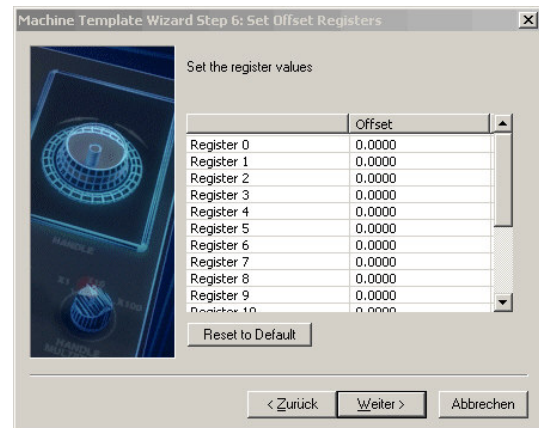
Schritt 3: Als Befehls-Vorlage die zuvor kopierte TTM-Datei suchen.
Hier: Luxmill.TTM im Ordner „Machines“.



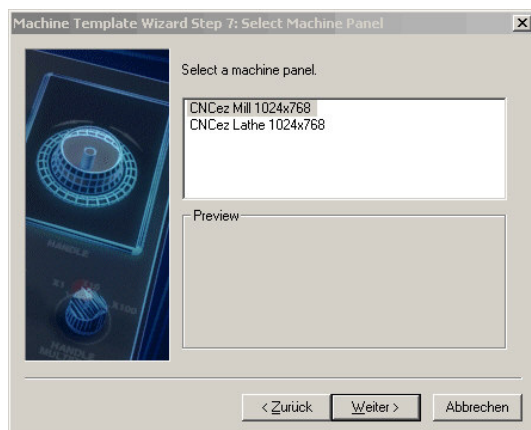
Schritt 4: Die Befehle überprüfen, wenn vorher nicht geändert, dann hier ändern.
Beispiel: G74 greift auf die Funktion G28 (Referenzpunktanfahren) zu.



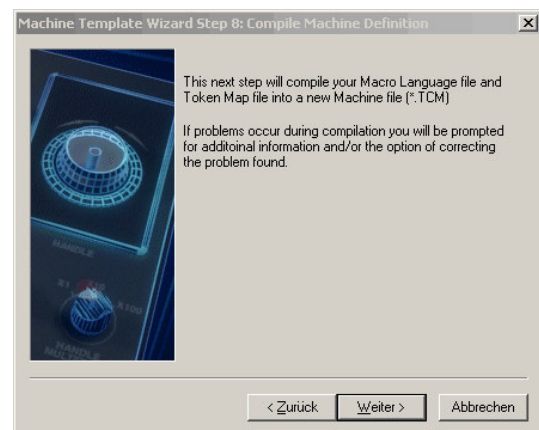
Schritt 5: Ihre Maschinenparameter eingeben



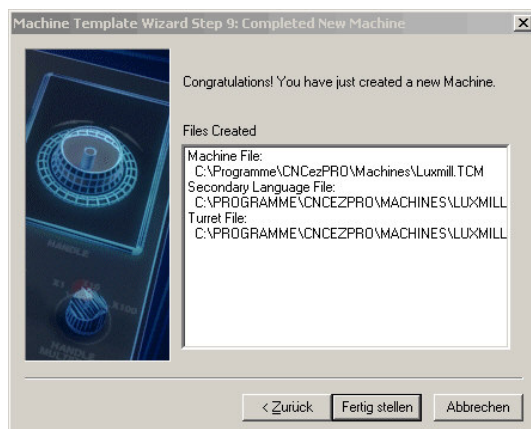
Schritt 6: alle Werkzeugkorrektur-Werte auf Null setzen



Schritt 7: Maschinen-Bedienfeld „Mill“ (Fräsmaschine) wählen



Schritt 8: fertig zum Kompilieren



Schritt 9: Es erscheint eine Glückwunschnmeldung, dass der Vorgang erfolgreich abgeschlossen wurde. CNCezPRO hat die TCM-, TSL- und TRT-Dateien erstellt. Falls bei der Programmierung in der Makrosprache Fehler gemacht wurden, erscheint stattdessen eine Fehlermeldung. Der Compiler beinhaltet eine Überprüfungs-funktion, welche Syntaxfehler (Rechtschreibfehler) und Plausibilitätsfehler (kann es sein, dass...?) erkennt.

Postprozessor für Traub anpassen

1. Auswahl des Referenz-Postprozessors nach Durchsicht der Funktionen in verschiedenen TML-Dateien. Hier wurde SampleLathe1 (entspricht Fanuc 6T) ausgewählt, weil dieser den Funktionen der Traub am nächsten kommt.
Kopieren der TML- und TTM-Datei des Referenz-Postprozessors SampleLathe1.TML und SampleLathe1.TTM vom Ordner „Machines“ in einen anderen Ordner.
Umbenennen der Dateien in Traub.TML und Traub.TTM
Zurückkopieren nach „Machines“.
2. Beim Vergleich der Befehlslisten fällt folgendes auf:
Traub benötigt A als Unterprogrammnummer bei Aufruf eines Unterprogramms, SampleLathe1 nimmt dazu P.
Für den Vorschub bei fallender Kontur benutzt Traub den Befehl E, bei SampleLathe1 ist dafür nur ein Integerwert vorgesehen.
Traub ruft die Unterprogramme mit G22 auf, SampleLathe1 mit M98.
Bei SampleLathe1 gibt es nur den Referenzpunkt G28, Traub bietet zusätzlich 4 Werkzeugwechsellpunkte von G24 bis G27 an.
Gewindedrehen ist bei Traub G33, Bei SampleLathe1 ist dies G32.
Traub kann Barrierenpunkte (ähnlich wie Arbeitsraumbegrenzung) mit G65 und G66 ein- und ausschalten.
Die Ausführung fallender Konturen in den Abspanzyklen kann bei Traub mit G88 und G89 ein- und ausgeschaltet werden.
Spindeldrehzahlbegrenzung G92 ist in SampleLathe1 nicht enthalten.
Umschaltung von Vorschub in mm oder Vorschub pro Umdrehung wird bei Traub mit G94 und G95 ausgeführt, bei SampleLathe1 mit G98 und G99.
Der Funktionsumfang dieser Maschine wurde ebenfalls um einige Zusatzoptionen erweitert, nämlich: M86 Bearbeitungstür öffnen, M87 Bearbeitungstür schließen.
V wird zur Eingabe der Schnittgeschwindigkeit benötigt, ist bei SampleMill1 nicht vorgesehen.
3. Wir ändern oder fügen die folgenden Zeilen der TTM-Datei hinzu:
A^4^404^10^0^SETP^0^ wir setzen bei A die Werte von P (für Unterprogrammnummer) ein.
E^2^404^10^7^SETE^0^ E ist normalerweise nur für Integerzahl (Ganzzahl) vorgesehen, deshalb war der ursprüngliche Typ-Parameter E^3^... wir ändern den Parameter in E^2^... (Dezimalzahl), da wir E für den Vorschub bei fallender Kontur in den Abspanzyklen benötigen.
Für den Aufruf von Unterprogrammen setzen wir bei G22 die Werte von M98 ein: G22^9^400^-10^0^m98^0^
Allen Werkzeugwechsellpunkten geben wir die Werte zum Referenzpunkt anfahren G28: G24^8^300^1^0^g28^0^ G25^8^300^1^0^g28^0^
G26^8^300^1^0^g28^0^ G27^8^300^1^0^g28^0^.
Wir sagen einfach, G33 sei wie G32, so dass Traub mit G33 Gewinde drehen kann, also: G33^9^301^-1^0^g32^0^
Barrierenpunkte können mit G65 und G66 nicht dargestellt werden, deshalb geben wir NOTHING ein: G65^9^300^1^0^NOTHING^0^
G66^9^300^1^0^NOTHING^0^

Fallende Konturen können bei SampleLathe1 nicht dargestellt werden, deshalb geben wir bei G88, G89 NOTHING ein:

G88^9^300^1^0^NOTHING^0^ G89^9^300^1^0^NOTHING^0^

Die Spindelhöchstdrehzahl G92 ist ebenfalls nicht darstellbar, deshalb: G92^9^300^1^0^NOTHING^0^

Das Umschalten für Vorschub in mm/min oder mm/U wird von G94 und G96 auf G98 und G99 geändert: G94^9^305^1^0^g98^0^ G95^9^304^1^0^g99^0^.
Damit V als Schnittgeschwindigkeit von SampleLathe1 erkannt wird, wird V zu S konvertiert: $V^2 \cdot 404 \cdot 10^4 \cdot s^0$

4. Diese Spezialfunktionen sollen der Vollständigkeit halber beispielhaft erwähnt werden, wenn Sie diese Optionen nicht haben, können Sie es weglassen
M86^9^400^-10^0^m00^0^ M87^9^400^-10^0^m00^0^. M86 und M87 steuern ein Ventil an, welches die Tür des Bearbeitungsraumes öffnet oder schließt. Weil das CNC-Programm vor dem Öffnen steht und vor dem Schließen auf eine Bestätigung (Startbefehl) wartet, setzen wir dafür die Werte von M00 ein.
5. Traub lässt die Definition der Verweilzeiten in X und U zu (in Sekunden). Dazu gehen wir in die Datei Traub.TML und suchen G04, es erscheint die Funktion G04, die wir wie untenstehend (rot) abändern:

```
/******  
  
FUNCTION g04()  
  
LOCAL  
    temp  
  
    clear_variables()  
  
    gcode:=4  
  
    u:=GETU()  
    x:=GETX()  
  
    IF (u==UNDEFINED AND x==UNDEFINED)  
        THEN  
            PROGRAMSTOP()  
        ELSE  
            IF (x!=UNDEFINED)  
                THEN  
                    temp:=x*1000  
                    DWELL(temp)  
                ELSE  
                    temp:=u*1000  
                    DWELL(temp)  
            ENDIF  
        ENDIF  
  
    RETURN x  
ENDFUNCTION  
  
/******
```

Die Abarbeitung der Verweilzeiten der Abspanzyklen G70, G71 und G72 steht jeweils in den Zeilen mit $d := d/1000$ oder $d := d/10000$.

```
d := d/1000          //necessary because system function EXECUTEPROFILE takes in "real values" - not thousandths  
d := d/10000        //necessary because system function EXECUTEPROFILE takes in "real values" - not thousandths
```

Diese Divisionen ergeben falsche Verweilzeiten, wir suchen sie ebenfalls in Traub.TML bei G70, G71 und G72 und kommentieren sie einfach aus, indem wir doppelte Schrägstriche davor setzen:

`//d := d/1000` oder `//d := d/10000`

Alternativ können auch die jeweiligen Zeilen gelöscht werden.

Nähere Erläuterungen zu Änderungen in der Makrosprache, siehe oben bei Luxmill.

Alle diese Änderungen sind schon in den beigegeführten Postprozessor-Dateien enthalten.

Anmerkungen zur Steuerung Traub:

Beim Gebrauch der Standard-Abspanzyklen G70 bis G72 muss die erste Satznummer der Konturbeschreibung (P) mindestens ab N50 beginnen, die kleineren sind reserviert, z. B. für die Zeilennummern bei einem Werkzeugwechsel.

Traub bietet verschiedene Zyklen mit zum Teil mächtigen Funktionen an, die verständlicherweise nur mit einem umfangreichen Ausbau des Makrosprache-Programms (TML-Datei) dargestellt werden könnten.

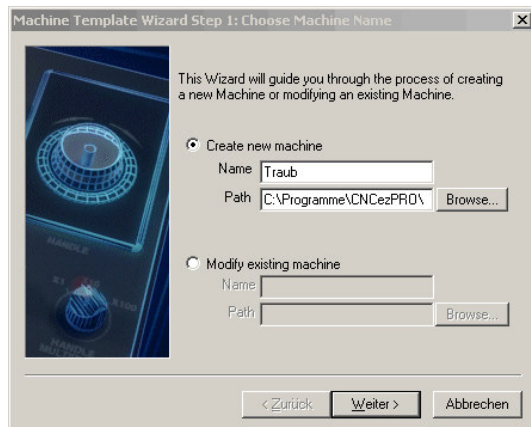
Auch die zusätzlichen Ausbaustufen, wie 2. Werkzeugrevolver, Gegenspindel, Rückseiten-Bearbeitung, Fräsfunktionen, usw. können mit dieser provisorischen Postprozessor-Anpassung nicht verwirklicht werden.

Für Vollständigkeit und Funktion kann keine Garantie übernommen werden.

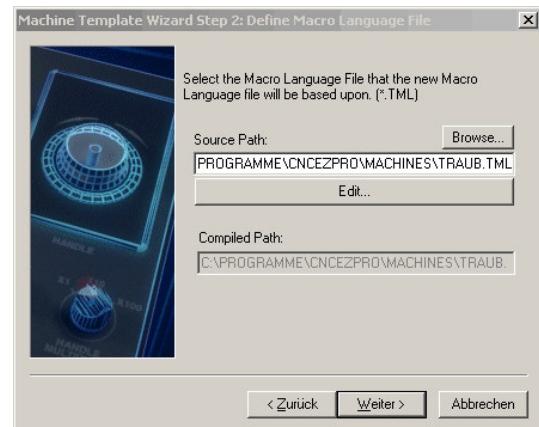
Der Autor haftet nicht für eventuelle Schäden.

Maschinen-Assistent benutzen und kompilieren - Traub

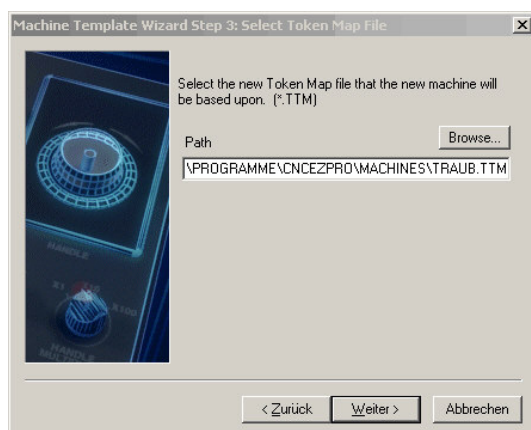
Im Menu Optionen > Maschinenassistent aufrufen



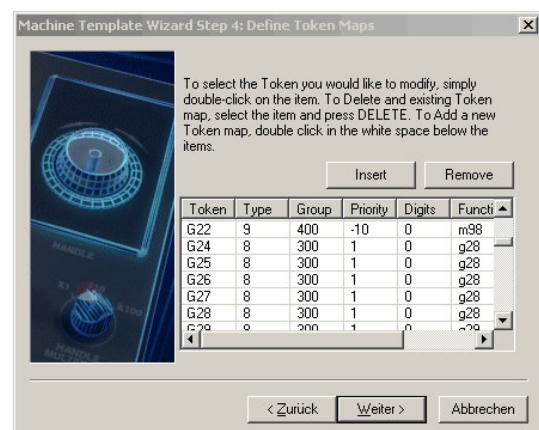
Schritt 1: Name der Maschine oder Steuerung eingeben.
Pfad wird automatisch vorgegeben, kann bei Bedarf geändert werden.



Schritt 2: Als Vorlage die zuvor kopierte TML-Datei suchen.
Hier: Traub.TML im Ordner „Machines“.



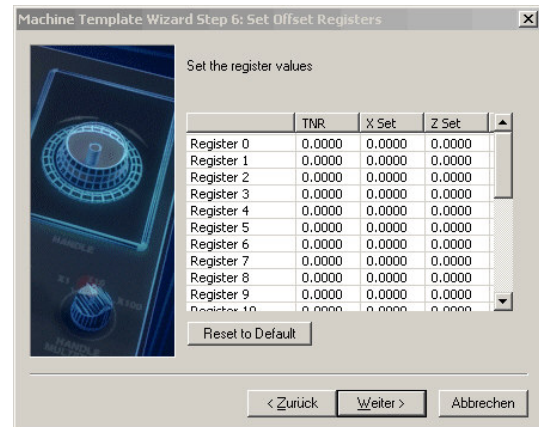
Schritt 3: Als Befehls-Vorlage die zuvor kopierte TTM-Datei suchen.
Hier: Traub.TTM im Ordner „Machines“.



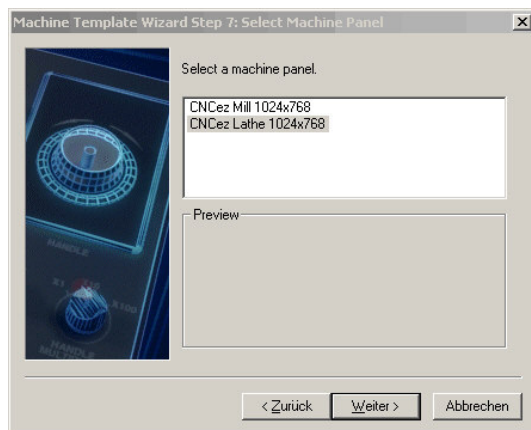
Schritt 4: Die Befehle überprüfen, wenn vorher nicht geändert, dann hier ändern.
Beispiele: G22 wird zu M98,
G24 bis G27 werden alle zu G28



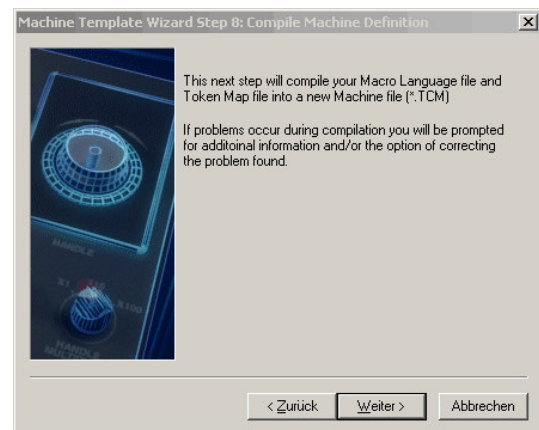
Schritt 5: Ihre Maschinenparameter eingeben



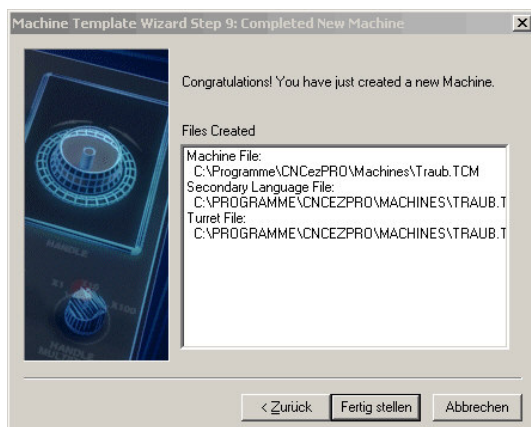
Schritt 6: alle Werkzeugkorrektur-Werte auf Null setzen



Schritt 7: Maschinen-Bedienfeld „Lathe“ (Drehmaschine) wählen



Schritt 8: fertig zum Kompilieren



Schritt 9: Es erscheint eine Glückwunschnmeldung, dass der Vorgang erfolgreich abgeschlossen wurde. CNCezPRO hat die TCM-, TSL- und TRT-Dateien erstellt. Falls bei der Programmierung in der Makrosprache Fehler gemacht wurden, erscheint stattdessen eine Fehlermeldung.