*LinuxCNC

automatischer Werkzeugwechsel

ATC

Version 2.8

Theorie – einfacher automatischer Werkzeugwechsel

Die Werkzeuge sollen automatisch aus einem Magazin geholt und wieder weggebracht werden.
Es gibt viele Möglichkeiten das zu lösen und eine davon werde ich hier aufzeigen.

Ich nehme mal folgendes an……..

Ein Magazin mit 20 Plätzen wird durch einen Schrittmotor positioniert.

Somit kann ich die Positionierung einfach im G-Code realisieren.
Außerdem wird die Achse zu beginn referenziert was einen klaren Ausgangspunkt ermöglicht.

Das Magazin hat eine Tür welche zu beginn geöffnet und am Ende wieder geschlossen werden muss.
Die Tür/Deckel hat einen Sensor der den Zustand verrät und ein einfach wirkendes Magnetventil.

Wenn Ausgang Ventil = 0 dann ist die Tür zu und der Sensor gibt eine 1 zur Kontrolle.
Wenn Ausgang Ventil = 1 dann ist die Tür offen und der Sensor gibt eine 0 zur Kontrolle.

Bei der Spannzange in der Spindel ist es genauso. (Werkzeugspannung)

Wenn Ausgang Ventil = 0 dann ist die Spannzange zu und der Sensor gibt eine 1 zur Kontrolle.
Wenn Ausgang Ventil = 1 dann ist die Spannzange offen und der Sensor gibt eine 0 zur Kontrolle.

Daraus ergibt sich das wir 2 Ausgänge und 2 Eingänge im Werkzeugwechselzyklus schalten bzw. abfragen müssen.
Wenn nun ein Werkzeug mit Tx M6 aufgerufen wird muss die Maschine eigenständig diese Aufgabe lösen können.

Theorie - einfacher automatischer Werkzeugwechsel

Meine Aufgaben und Lösungsansätze……..

Ich erstelle dazu Unterprogramme (NC) die beim Werkzeugaufruf abgearbeitet werden.
- Dazu muss ich aber auch den eigentlichen M6 Befehl übergehen damit mein Unterprogramm zuerst ausgeführt wird.

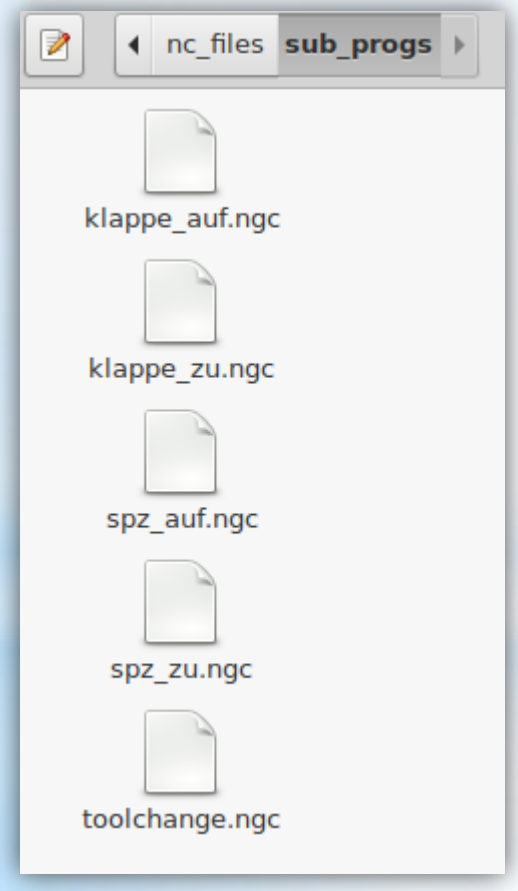
Aus den NC Programmen heraus sollen die Eingänge abgefragt und Ausgänge geschalten werden.
- Dies kann ich mit dem im Modul „motion“ zur Verfügung stehenden digital-in und digital-out Pins machen.

Außerdem soll das Werkzeug über einen Neustart der Maschine nicht verloren gehen.
- Hier speichere ich bei jeden Werkzeugwechsel den Wert und frage diesen immer im Unterprogramm ab.
- Das verhindert ein doppeltes einwechseln von Werkzeugen und nach einem Neustart kann man ohne Angst einen Wechsel durchführen

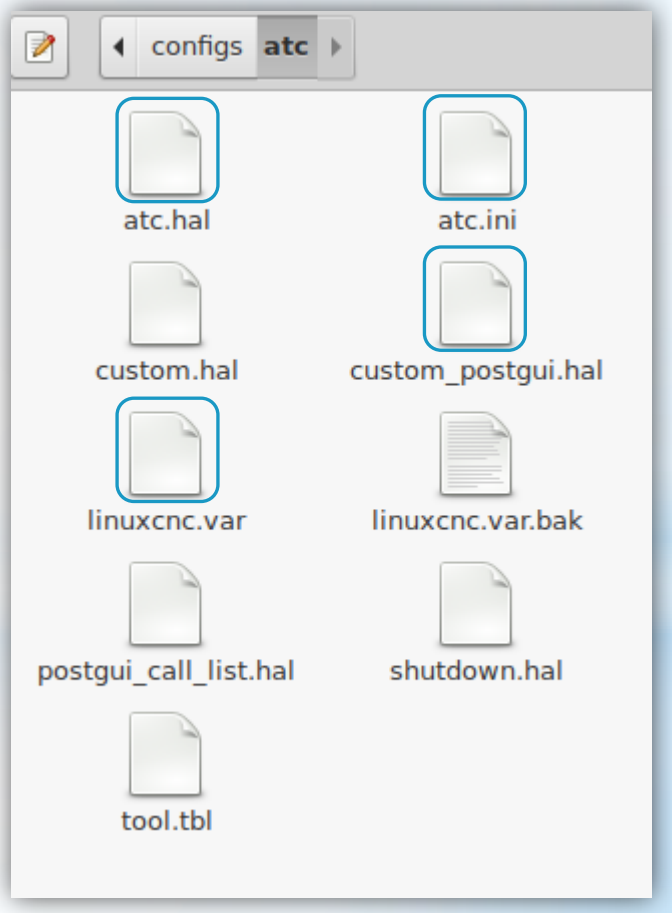
Beim Befehl T0 M6 soll das aktuelle Werkzeug weggebracht werden so das die Spindel am Ende leer ist.
- Dies muss ich durch „if“ „else“ Abfragen in den Unterprogrammen realisieren.

Übersicht an welchen Orten ich arbeiten muss.

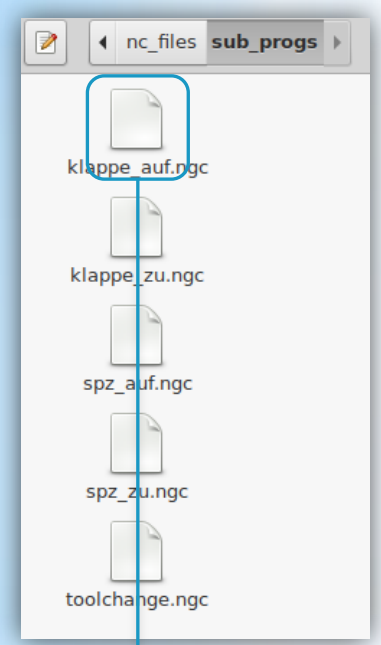
Für die Unterprogramm habe ich unter „nc_files“ einen Ordner mit dem Namen „sub_progs“ angelegt.
Darin liegen meine wichtigen Unterprogramm.



Meine Testmaschine heißt atc .
Im Maschinenordner werde ich die markierten Dateien anpassen.



Unterprogramme



Unterprogramm beginnen immer mit dem Buchstaben „o“
Danach folgt der Name und ggf. mit Pfadangabe.
Das Wörtchen „sub“ und „endsub“ kennzeichnen den Beginn und das Ende.

Das Unterprogramm wird im G-Code Style geschrieben.
Hier natürlich mit Parametern, Ausgabemeldung und Abfragen.

Informationen zur Programmierung gibt’s hier.
http://linuxcnc.org/docs/2.7/html/
http://linuxcnc.org/docs/2.7/html/gcode/o-code.html

Das Programm „klappe_auf“ setzt den Digitalausgang P0 und wartet dann auf den Digitaleingang P0. Wenn dieser nicht innerhalb von 20 Sekunden auf high geht springt es in die o100 if Schleife. Dort wird einerseits die Meldung und ein M0 Stop generiert.
Ansonsten läuft das Programm durch und wird mit m2 beendet.

Das mal zum Aufbau der Unterprogramme.

klappe_auf.ngc

```
o<sub_progs/klappe_auf> sub

M64 P0                      (Ausgang setzen fü Klappe auf)
M66 P0 L3 Q#4000            (Eingang ob Klappe offen ?)
o100 if [#5399 EQ -1]       (Stop falls Klappe nicht öffnet. M0)
(DEBUG,STOP: Klappe nicht offen!)
(DEBUG,Klappe manuell öffen dann weiter mit Start oder Programm abbrechen! Crash Gefahr!)
M0                          (Porgrammstop, mit Start geht es weiter)
o100 endif

o<sub_progs/klappe_auf> endsub
m2
```

Unterprogramm toolchange.ngc

Im Unterprogramm toolchange.ngc habe ich alle Fahrbewegungen , Abfragen, Aufrufe und Entscheidungswege geschrieben.

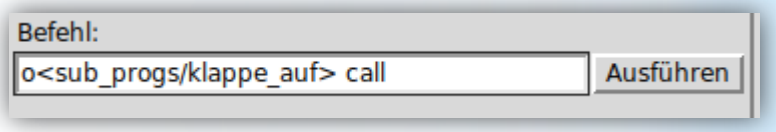
toolchange.ngc

```
o<sub_progs/toolchange> sub
#4001=18                                    (bei 20 Taschen 360 Grad durch 20)
#4000=10                                    (Wartezeit in Sekunden für die Eingänge)
M61 Q#4999                                  (Werkzeug aus Speicher setzen)
o500 if[#<_current_tool> EQ #<_selected_tool>]
(DEBUG,Tool bereits in Spindel)
o500 elseif [#<_current_tool> NE #<_selected_tool>]
G0 G90 G53 X100 Y0 Z50 M19 R0               (Spindelpositionierung Bnd Orientierung)
o501 if[#<_current_tool> EQ 0]              (aktuelles WZ ist T0)
    o<sub_progs/klappe_auf> call
    G53 Z150                                (ohne WZ einfahren)
    G53 X200                                (ohne WZ einfahren)
    G53 X300                                (Endposition)
    G0 B[#<_selected_tool> * #4001-#4001]   (Karusell drehen)
    o<sub_progs/spz_auf> call
    G53 Z50                                 (wieder runter neues wz)
    o<sub_progs/spz_zu> call
    M6 G43
    #4999=#5400                             (WZ dauerhaft merken)
    G53 X100                                (mit WZ ausfahren)
    G53 Z50
    o<sub_progs/klappe_zu> call
    G53 X0

o501 elseif [#<_current_tool> GT 0]         (aktuelles WZ nicht T0)
    o502 if [#<_selected_tool> EQ 0]        (gewähltes WZ T0)
        G0 B[#<_current_tool> * #4001-#4001] (Karusell drehen)
        o<sub_progs/klappe_auf> call
        G53 X200                            (Mit WZ einfahren)
        G53 X300                            (Endposition)
        o<sub_progs/spz_auf> call
        G53 Z150                            (hochfahren)
        o<sub_progs/spz_zu> call
        M6 G43
        #4999=#5400                         (WZ dauerhaft merken)
        G53 X100
        o<sub_progs/klappe_zu> call
        G53 Z50
        G53 X0
    o502 elseif [#<_selected_tool> NE 0]    (gewähltes WZ nicht T0)
        G0 B[#<_current_tool> * #4001-#4001] (Karusell drehen)
        o<sub_progs/klappe_auf> call
        G53 X200                            (Mit WZ einfahren)
        G53 X300                            (Endposition)
        o<sub_progs/spz_auf> call
        G53 Z150                            (hochfahren)
        G0 B[#<_selected_tool> * #4001-#4001] (Karusell drehen)
        G53 Z50                             (wieder runter neues wz)
        o<sub_progs/spz_zu> call
        M6 G43
        #4999=#5400                         (WZ dauerhaft merken)
        G53 X100
        G53 Z50
        o<sub_progs/klappe_zu> call
        G53 X0
    o502 endif
o501 endif
o500 endif
o<sub_progs/toolchange> endsub
m2
```

Eigentlich kann ich die Unterprogramme „klappe“ und „spz“ auch direkt hier rein programmieren.

Ich hab mich aber dagegen entschieden.
Denn die Unterprogramme kann ich somit auch mal einzeln verwenden.

Zum Beispiel im MDI mal die Klappe öffnen oder Schließen.



Maschinendatei - atc.ini

In der Sektion [RS274NGC] habe ich die Zeile „REMAP = M6 modalgoup=6 ngc=sub_progs/toolchange“ ergänzt.
Dies bewirkt das beim M6 Befehl das Unterprogramm „toolchange.ngc“ ausgeführt wird.

```
[RS274NGC]
PARAMETER_FILE = linuxcnc.var

REMAP= M6 modalgroup=6 ngc=sub_progs/toolchange
```

In der Sektion [KINS] und [TRAJ] habe ich XYZ mit B ergänzt und JOINTS auf 4 geändert.
B wird meine Rotationsachse für das Magazin.

```
[KINS]
JOINTS = 4
KINEMATICS = trivkins coordinates=XYZB

[TRAJ]
COORDINATES =  XYZB
LINEAR_UNITS = mm
ANGULAR_UNITS = degree
```

Weiter habe ich eine neue Sektion [AXIS_B] und [JOINT_3] mit folgenden Werten hinzugefügt.
Wie auch für eine Normale Achse nur mit dem Unterschied das ich diese als ANGULAR laufen lasse.

```
#********************************************
[AXIS_B]
MAX_VELOCITY = 66
MAX_ACCELERATION = 1000
MIN_LIMIT = -0
MAX_LIMIT = 360

[JOINT_3]

TYPE = ANGULAR
WRAPPED_ROTARY = 1
HOME = 0.0
FERROR = 10.0
MIN_FERROR = 1.0
MAX_VELOCITY = 66
MAX_ACCELERATION = 1000
STEPGEN_MAXVEL = 100
STEPGEN_MAXACCEL = 2000

# these are in nanoseconds
DIRSETUP   = 5000
DIRHOLD    = 5000
STEPLEN    = 5000
STEPSPACE  = 5000
STEP_SCALE = 1
MIN_LIMIT = -0
MAX_LIMIT = 360
HOME_OFFSET = 0.0
HOME_SEQUENCE = 1
```

Maschinendatei - atc.hal

Damit die B-Achse auch reagiert habe ich in der HAL den „stepgen.0“ mit „joint.3“ verknüpft. Bzw. folgenden Block hinzugefügt.

```
#********************
#  AXIS B JOINT 3
#********************

# Step Gen signals/setup

setp   hm2_7i92.0.stepgen.00.dirsetup        [JOINT_3]DIRSETUP
setp   hm2_7i92.0.stepgen.00.dirhold         [JOINT_3]DIRHOLD
setp   hm2_7i92.0.stepgen.00.steplen         [JOINT_3]STEPLEN
setp   hm2_7i92.0.stepgen.00.stepspace       [JOINT_3]STEPSPACE
setp   hm2_7i92.0.stepgen.00.position-scale  [JOINT_3]STEP_SCALE
setp   hm2_7i92.0.stepgen.00.step_type        0
setp   hm2_7i92.0.stepgen.00.control-type     0
setp   hm2_7i92.0.stepgen.00.maxaccel         [JOINT_3]STEPGEN_MAXACCEL
setp   hm2_7i92.0.stepgen.00.maxvel           [JOINT_3]STEPGEN_MAXVEL

net x-pos-cmd    joint.3.motor-pos-cmd => hm2_7i92.0.stepgen.00.position-cmd
net x-pos-fb     joint.3.motor-pos-fb  <= hm2_7i92.0.stepgen.00.position-fb
net x-enable     joint.3.amp-enable-out => hm2_7i92.0.stepgen.00.enable
```

Am Ende der HAL wird der manuelle Werkzeugwechsel so nicht mehr benötigt.

```
#  ---manual tool change signals---

loadusr -W hal_manualtoolchange
net tool-change-request     iocontrol.0.tool-change       =>  hal_manualtoolchange.change
net tool-change-confirmed   iocontrol.0.tool-changed      <=  hal_manualtoolchange.changed
net tool-number             iocontrol.0.tool-prep-number  =>  hal_manualtoolchange.number
net tool-prepare-loopback   iocontrol.0.tool-prepare      =>  iocontrol.0.tool-prepared
```

Daher habe ich den Block auf folgendes reduziert.

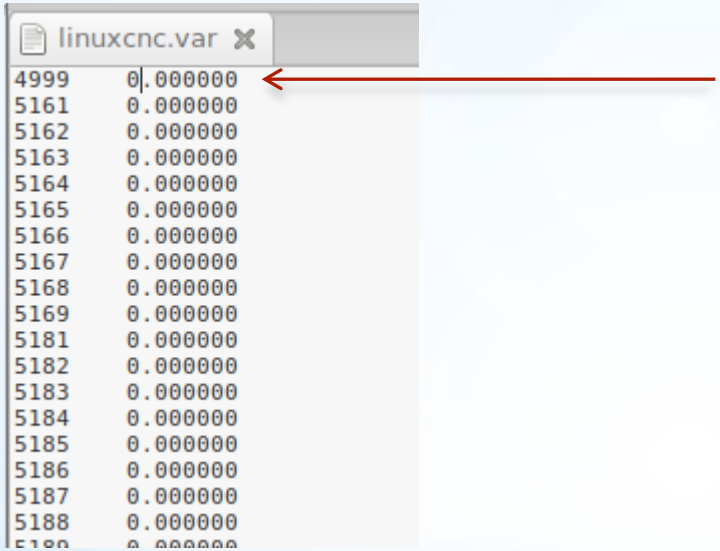
```
#  ---manual tool change signals---

net tool-change-confirmed   iocontrol.0.tool-changed      <=  iocontrol.0.tool-change
net tool-prepare-loopback   iocontrol.0.tool-prepare      =>  iocontrol.0.tool-prepared
```

Maschinendatei - linuxcnc.var

In der linuxcnc.var habe ich ganz oben den Parameter #4999 hinzugefügt.

Dort wird dann bei jeden Werkzeugwechsel die aktuelle Werkzeugnummer gespeichert.

Im falle eines Neustart von LinuxCNC kann so sichergestellt werden das der ATC keinen Crash fährt ☺



Maschinendatei - custom_postgui.hal

Nun muss ich die digitalen Ein und Ausgänge, welche in den Unterprogramme bearbeitet werden, mit den wirklichen Ein und Ausgänge verknüpfen. Damit die Ventile schalten und die Sensorergebnisse im Unterprogramm ankommen.

In meinen Testaufbau nutze ich IOs von der 7i92. Die Ausgänge muss ich vorher auch als Ausgänge deklarieren und die Eingänge frage ich negiert also mit „in_not“ ab.

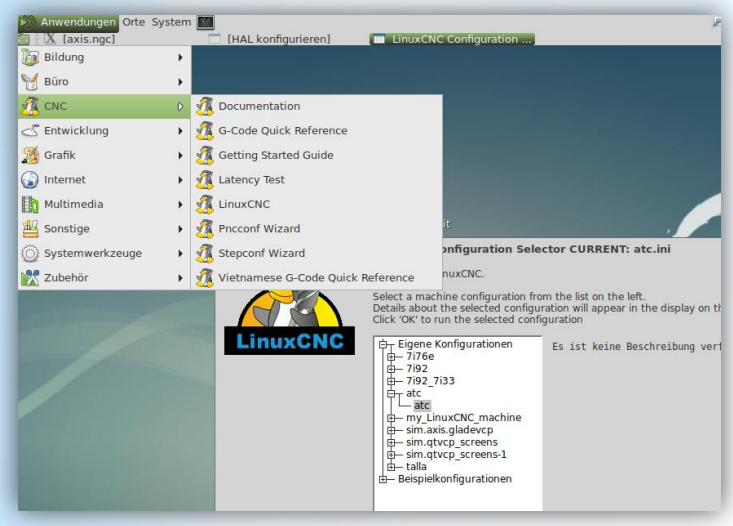
Die Netznamen habe ich so gewählt das ich die Funktion erkennen kann.

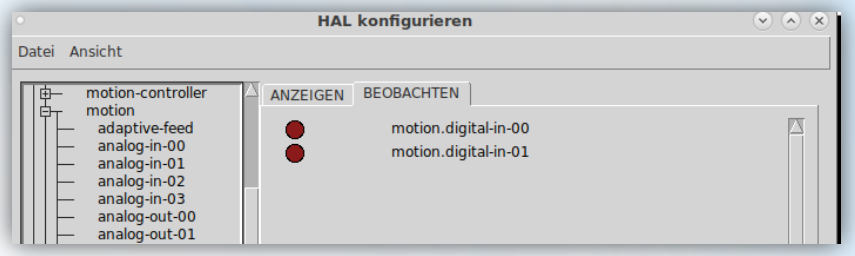
custom_postgui.hal

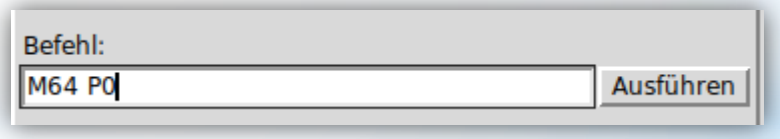
```
# Include your custom_postgui HAL commands here
# This file will not be overwritten when you run PNCconf again


setp hm2_7i92.0.gpio.018.is_output true
setp hm2_7i92.0.gpio.019.is_output true

net ventil-klappe  hm2_7i92.0.gpio.018.out <= motion.digital-out-00
net ventil-spindel hm2_7i92.0.gpio.019.out <= motion.digital-out-01

net ini-klappe     hm2_7i92.0.gpio.020.in_not <= motion.digital-in-00
net ini-spz        hm2_7i92.0.gpio.021.in_not <= motion.digital-in-01
```

Starten und testen

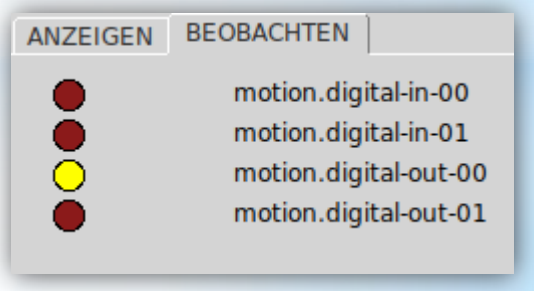
Wenn man sich die PINs motion.digital-in-00 und motion.digital-in-01 im Beobachten anzeigen lässt, dann müssen dort die Sensorergebnisse auftauchen.
Sprich wenn man die Sensoren auslöst muss sich der Zustand ändern.







Das gleiche kann man auch mit den Ausgängen.
Wenn man im MDI den Befehl „M64 P0“ ausführt muss der erste Ausgang auf high gehen. Mit „M65 P0“ geht er wieder low.



Meine Dateien für den einfachen automatischen Werkzeugwechsel findet ihr wieder unter….

http://talla83.de/linuxcnc/config.htm

„erzeugt für Version 2.8“

*LinuxCNC