LinuxCNC V2.10.0~pre2

2021-10-28

Table of Contents

Erste Schritte & Konfiguration . . . 1
1. Erste Schritte mit LinuxCNC . . . 2
1.1. Über LinuxCNC . . . 2
1.2. System-Anforderungen . . . 6
1.3.Hardware-Schnittstelle . . . 8
1.4. Beziehen von LinuxCNC . . . 13
1.5. LinuxCNC ausführen . . . 20
1.6. LinuxCNC updaten . . . 25
1.7. Linux FAQ . . . 34
2. Allgemeine Informationen für Anwender . . . 40
2.1. Vorwort für Anwender . . . 40
2.2. LinuxCNC Anwender-Einführung . . . 41
2.3. Wichtige Anwendungskonzepte . . . 52
2.4. LinuxCNC starten . . . 58
2.5. CNC-Maschinenübersicht . . . 60
2.6. Drehmaschinen Anwender-Information . . . 67
2.7. Einführung zum Plasmaschneiden LinuxCNC Benutzer . . . 77
3. Konfigurationsassistenten . . . 100
3.1. Schrittmotor (engl. stepper) Konfigurationsassistent . . . 100
3.2. Mesa-Konfigurationsassistent . . . 115
4. Konfiguration . . . 141
4.1. Integrator-Konzepte . . . 141
4.2. Latenz-Test . . . 148
4.3. Stepper-Abstimmung . . . 154
4.4.INI-Konfiguration . . . 158
4.5. MTConnect . . . 193
4.6. Konfiguration der Referenzfahrt (engl. homing) . . . 197
4.7. Konfiguration der Drehmaschine . . . 207
4.8. Stepper Schnellstart . . . 209
4.9. Schrittmotor-Konfiguration . . . 212
4.10. Schrittmotor Diagnostik . . . 216
4.11. Filter-Programme . . . 219
5. HAL (Hardware Abstraction Layer) . . . 223
5.1. HAL Einführung . . . 223
5.2. HAL Grundlagen . . . 233
5.3. HAL TWOPASS . . . 245
5.4. HAL Werkzeuge (engl. tools) . . . 249
5.5. HAL Tutorial . . . 257
5.6. HAL Beispiele . . . 291
5.7. Kern-Komponenten . . . 297

5.8. HAL-Komponenten Liste . . . 305
5.9. Beschreibungen der HAL-Komponenten . . . 325
5.10. HAL Komponenten Erstellung . . . 346
5.11. HALTCL-Dateien . . . 363
5.12. HAL-Benutzeroberfläche . . . 367
5.13. Halui Beispiele . . . 375
5.14. Anlegen von Nicht-Echtzeit Python Komponenten . . . 377
5.15. Kanonische Device Schnittstelle . . . 381
6. Hardware-Treiber . . . 384
6.1. Parallelport-Treiber . . . 384
6.2. General Mechatronics Treiber . . . 390
6.3. GS2 VFD-Treiber . . . 412
6.4. HAL Treiber für Raspberry Pi GPIO-Pins . . . 414
6.5. Generische Treiber für alle GPIO unterstützt von gpiod. . . . 417
6.6. Mesa HostMot2-Treiber . . . 420
6.7. MB2HAL . . . 435
6.8. Mitsub VFD-Treiber . . . 446
6.9. Pico-Treiber . . . 449
6.10. Powermax Modbus-Treiber . . . 453
6.11. Shuttle . . . 455
6.12. VFS11 VFD-Treiber . . . 456
7. Hardware-Beispiele . . . 463
7.1. PCI-Parallelport . . . 463
7.2. Spindelsteuerung . . . 463
7.3. MPG Handgeräte . . . 468
7.4. GS2 Spindel . . . 470
8. ClassicLadder . . . 472
8.1. ClassicLadder Einführung . . . 472
8.2. ClassicLadder Programmierung . . . 475
8.3. ClassicLadder Beispiele . . . 509
9. Fortgeschrittene Themen . . . 515
9.1. Kinematik . . . 515
9.2. Kinematics Conventions . . . 521
9.3. Einrichten "modifizierter" Denavit-Hartenberg (DH)-Parameter für "genserkins" . . . 530
9.4. 5-Achsen Kinematik . . . 551
9.5. Schaltbare Kinematik (switchkins) . . . 567
9.6. PID-Feineinstellung (engl. tuning) . . . 578
9.7. Neuzuordnung (engl. remap) für das Erweitern von G-Code . . . 584
9.8. Moveoff-Komponente . . . 635
9.9. Eigenständiger Interpreter . . . 640
9.10. Offsets der externen Achse . . . 641
9.11. Werkzeugdatenbank-Schnittstelle . . . 647

Anwendung . . . 653
10. Benutzerschnittstellen . . . 654
10.1. AXIS GUI . . . 654
10.2. GMOCCAPY . . . 687
10.3. Die Touchy Grafische Benutzeroberfläche . . . 732
10.4. Gscreen . . . 736
10.5. QtDragon GUI . . . 751
10.6. NGCGUI . . . 797
10.7. TkLinuxCNC GUI . . . 817
10.8. QtPlasmaC . . . 823
10.9. MDRO GUI . . . 962
11. G-Code Programmierung . . . 966
11.1. Koordinatensysteme . . . 966
11.2. Werkzeug Kompensation . . . 974
11.3. GUI zur Werkzeug-Bearbeitung . . . 984
11.4. Überblick zur G-Code Programmierung . . . 987
11.5. G-Codes . . . 1016
11.6. M-Codes . . . 1081
11.7.O Codes . . . 1098
11.8. Andere Codes . . . 1107
11.9. G-Code Beispiele . . . 1109
11.10. Vom Bild zu G-Code . . . 1111
11.11. RS274/NGC Unterschiede . . . 1116
12. Virtuelle Schalttafeln . . . 1119
12.1. PyVCP . . . 1119
12.2. PyVCP-Beispiele . . . 1143
12.3. GladeVCP: Glade Virtuelle Schalttafel . . . 1155
12.4. GladeVCP-Bibliotheksmodule . . . 1226
12.5. GladeVCP mitgelieferte Panels . . . 1231
12.6. QtVCP . . . 1235
12.7. QtVCP Virtuelle Kontrollpanels . . . 1266
12.8. QtVCP Widgets . . . 1284
12.9. QtVCP-Bibliotheksmodule . . . 1368
12.10. QtVismach . . . 1399
12.11. QtVCP: Erstellung benutzerdefinierter Widgets . . . 1413
12.12. Code-Schnipsel aus der QtVCP-Handler-Datei . . . 1430
12.13. QtVCP-Entwicklung . . . 1451
13. Programmierung der Benutzeroberfläche . . . 1456
13.1. Panelui . . . 1456
13.2. Das LinuxCNC Python-Modul . . . 1464
13.3. Das HAL Python-Modul . . . 1487
13.4. Python module for managing the realtime backend . . . 1505

13.5. GStat Python-Modul 1505
13.6. Vismach 1520
Glossar, Copyright & Geschichte 1527
14. Umschlagseite 1528
15. Glossar 1529
16. Copyright 1537
16.1. Juristischer Abschnitt 1537
17. LinuxCNC Geschichte 1543
17.1. Origin (engl. für Ursprung) 1543

Erste Schritte & Konfiguration

Chapter 1. Erste Schritte mit LinuxCNC

1.1. Über LinuxCNC

LinuxCNC (the Enhanced Machine Control) is a software system for computer control of machine tools such as milling machines and lathes, robots such as puma and scara and other computer controlled machines up to 9 axes. LinuxCNC is free software with open source code. Current versions of LinuxCNC are entirely licensed under the GNU General Public License and Lesser GNU General Public License (GPL and LGPL).

To lower the entry-hurdle, LinuxCNC provides:

- einfaches Entdecken und Testen ohne Installation mit dem Live-Image,
- einfache Installation von der Live-Image,
- benutzerfreundliche grafische Konfigurationsassistenten zum schnellen Erstellen einer maschinenspezifischen Konfiguration,
- directly availability as regular packages of recent releases of Debian (since Bookworm) and Ubuntu (since Kinetic Kudu).

LinuxCNC provides a graphical user interface with many flavours to choose from to match your personal preferences and technical needs. Advanced users may directly exploit

- Tool zur Erstellung einer grafischen Benutzeroberfläche (Glade, Qt),
- the interpreter for *G-code* (the RS-274 machine tool programming language),
- Betrieb von Low-Level-Maschinenelektronik wie Sensoren und Motorantriebe,
- eine einfach zu bedienende *Steckplatinen*-Schicht für die schnelle Erstellung einer einzigartigen Konfiguration für Ihre Maschine,
- eine mit Leiterdiagrammen programmierbare Software-SPS.

Under the hood, LinuxCNC provides

- ein System zur Bewegungsplanung in Echtzeit mit Vorausschau,
- support for non-Cartesian motion systems is provided via custom kinematics modules. Available architectures include hexapods (Stewart platforms and similar concepts) and systems with rotary joints to provide motion such as PUMA or SCARA robots.
- support for a variety of hardware interfaces. The control can operate true servos (analog or PWM) with the feedback loop closed by the LinuxCNC software at the computer, or open loop with step-servos or stepper motors.
- Zu den Funktionen der Bewegungssteuerung gehören: Fräserradius- und Längenkompensation, auf eine bestimmte Toleranz begrenzte Bahnabweichung, Gewindedrehen, synchronisierte Achsenbewegung, adaptiver Vorschub, Vorschubübersteuerung durch den Bediener und konstante Geschwindigkeitsregelung.
- LinuxCNC runs on Linux using real-time extensions.

LinuxCNC expects G-code that if not entered manually is provided by another software, which supports CAM (Computer Automated Manufacturing) and determines what tool shall be used at what speed for what geometry. Many prominent CAD (Computer Automated Design) tools that determine the desired final shape of your work piece (or the assembly of multiple work pieces that area to be produced individually) offer a CAM module.

1.1.1. Architecture - Context diagram

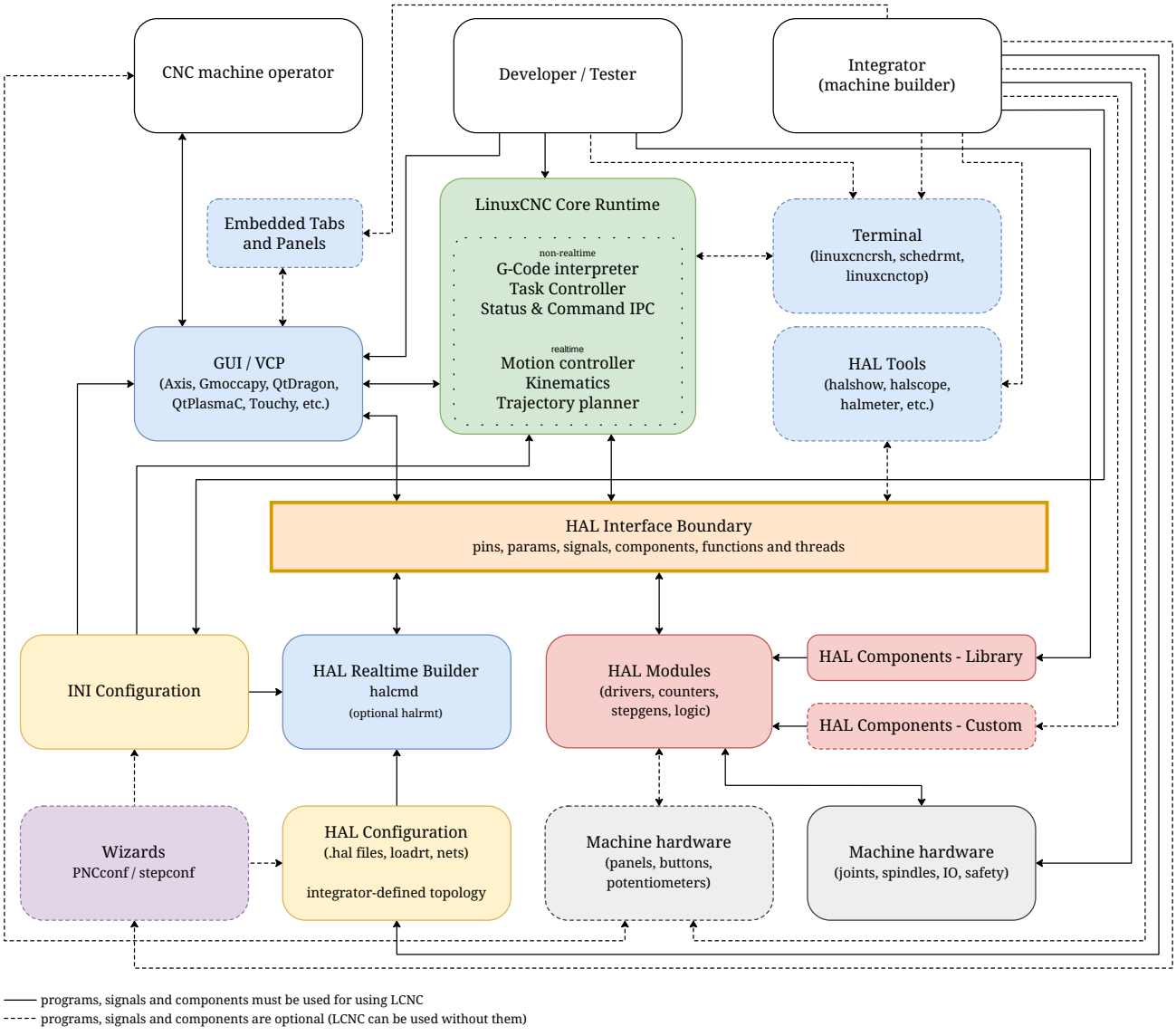


Figure 1. Roles of operators, integrators, developers and hardware

The diagram presents the components and players of the LinuxCNC ecosystem and how they interact. It is not intended to help you understand the functionality of LinuxCNC. Please refer to the following chapters for this.

Operator

One a machine is set up, its operator will only use one of the many graphical user interfaces that LinuxCNC and external groups are providing. The requirements for the operator are determined by how the integrator has set up the machine. The integrator has the option of setting up the machine so that the operator only presses one button to start the machining process, or leaves the GUI in its

default state and the operator will fully control the CNC machine using the GUI functionality and G,M,O-codes. The integrator may or may not create a physical or virtual panel for the operator with various buttons and various indicators.

Integrator

It is on an integrator (machine builder) to ensure that the LinuxCNC configuration matches the hardware setup both in the wiring and the protocols spoken on those wires. The integrator can choose whether to set up the machine using the Wizard or to configure it manually. If the Wizard is used, the integrator's knowledge of LinuxCNC is minimal. It is enough to understand the machine hardware. If the integrator wants to use the maximum potential of LinuxCNC, he must be able to create or edit configuration files manually. To do this, it is enough to have knowledge of HAL, INI configuration and ideally the creation of custom HAL components or embedded panels. This knowledge will allow the connection of various hardware combinations with LCNC. Using INI, the integrator selects the GUI (Gmoccapy, Axis, Qt, ...), kinematics, number of axes, parameters (velocities, acceleration, distance, ...). Using HAL, the integrator selects the hardware control method (velocity mode / position mode, on-off control / analog control, without / with feedback, ...). Using a suitable HAL module, various components can be controlled via various buses (PCI, USB, Ethernet, EtherCAT, Modbus RTU/TCP, Parallel port, ...)

Developer

The LinuxCNC developers may be coming up with drivers for new hardware or other new features in the GUI and anything in between a mouse click and a motor turning. For testing, monitoring or possibly also the communication between multiple machines, also a text-based interface to LinuxCNC is available. Since LinuxCNC is an Open-source project, you can modify it in any way you like, provided you meet the very benevolent license conditions. You can create these modifications for the official LinuxCNC community, or for your own needs. Both paths have their advantages and disadvantages. If you offer your modification or improvement to the official developers, if they are interested, they can help you improve it even more and you will receive feedback. If you keep your modification to yourself, you do not have to worry about whether it will interest the official developers, but it may be a problem in the future if someone unfamiliar with these modifications were to maintain the machine you built (modifications, updates, fixes, ...). Of course, the developers modify all the code that is part of LinuxCNC, but the diagram only shows the links for which the developer's skills are necessary (C, C++, Python, Bash, GTK, Glade, QT, Linux OS, GitHub, PC hardware, ...)

Wizard

Wizards are standalone programs that LinuxCNC and external groups are providing. They can work without other LinuxCNC components. The main output of Wizards are configuration files (*.ini, *.hal and others). Therefore, it is possible to do your first machine setup using the Wizard and only later, after a deeper study of the LCNC configuration, can you edit the files generated by the Wizard.

1.1.2. Das Betriebssystem

LinuxCNC ist als gebrauchsfertige Pakete für Debian-Distributionen verfügbar.

1.1.3. Wo bekomme ich Hilfe

Web-Forum

Ein Webforum finden Sie unter https://forum.linuxcnc.org oder indem Sie dem Link oben auf der linuxcnc.org Homepage folgen.

Diese ist recht aktiv, aber die Zielgruppe ist stärker auf die Benutzer ausgerichtet als die Mailingliste. Wenn Sie sicher sein wollen, dass Ihre Nachricht von den Entwicklern gesehen wird, sollten Sie die Mailingliste bevorzugen.

IRC

IRC steht für Internet Relay Chat. Es ist eine Live-Verbindung zu anderen LinuxCNC-Benutzern. Der LinuxCNC IRC-Kanal ist #linuxcnc auf libera.chat.

Der einfachste Weg, um ins IRC zu gelangen, ist die Verwendung des eingebetteten Webclient-Clients von libera.

Etwas IRC-Etikette

- Stellen Sie gezielte Fragen... Vermeiden Sie Fragen wie „Kann mir jemand helfen?“.
- Wenn Sie wirklich neu auf diesem Gebiet sind, denken Sie ein wenig über Ihre Frage nach, bevor Sie sie tippen. Stellen Sie sicher, dass Sie genügend Informationen geben, damit jemand Ihre Frage beantworten oder Ihr Problem lösen kann.
- Haben Sie etwas Geduld, wenn Sie auf eine Antwort warten. Manchmal dauert es eine Weile, bis eine Antwort formuliert wird, oder alle sind mit der Arbeit beschäftigt oder so.
- Richten Sie Ihr IRC-Konto mit Ihrem eindeutigen Namen ein, damit andere wissen, wer Sie sind. Wenn Sie den Java-Client verwenden, sollten Sie jedes Mal, wenn Sie sich anmelden, denselben Namen verwenden. So können sich die Leute merken, wer Sie sind, und wenn Sie schon einmal dabei waren, werden sich viele an die vergangenen Diskussionen erinnern, was für beide Seiten Zeit spart.

Dateien teilen

Die gängigste Art, Dateien im IRC auszutauschen, besteht darin, die Datei auf einen der folgenden oder einen ähnlichen Dienst hochzuladen und den Link einzufügen:

- *Für Text*: https://pastebin.com/, https://gist.github.com/, https://0bin.net/, https://paste.debian.net/
- *Für Bilder*: https://imagebin.org/, https://imgur.com/, https://bayimg.com/
- *Für Dateien*: https://filedropper.com/, https://filefactory.com/, https://1fichier.com/

Mailingliste

Eine Internet-Mailingliste ist eine Möglichkeit, Fragen zu stellen, die jeder auf dieser Liste sehen und nach Belieben beantworten kann. Auf einer Mailingliste können Sie Ihre Fragen besser stellen als im IRC, aber die Antworten dauern länger. Kurz gesagt: Sie senden eine Nachricht an die Liste und erhalten entweder tägliche Zusammenfassungen oder individuelle Antworten, je nachdem, wie Sie Ihr Konto

eingerichtet haben.

Sie können die Mailingliste emc-users abonnieren unter: https://lists.sourceforge.net/lists/listinfo/emc-users.

Web-Forum

Ein Webforum finden Sie unter https://forum.linuxcnc.org/ oder indem Sie dem Link oben auf der https://linuxcnc.org/ Homepage folgen.

Diese ist recht aktiv, aber die Zielgruppe ist stärker auf die Benutzer ausgerichtet als die Mailingliste. Wenn Sie sicher sein wollen, dass Ihre Nachricht von den Entwicklern gesehen wird, sollten Sie die Mailingliste bevorzugen.

LinuxCNC-Wiki

Eine Wiki-Site ist eine von Benutzern gepflegte Website, die von jedermann ergänzt und bearbeitet werden kann.

Die vom Benutzer gepflegte LinuxCNC-Wiki-Seite enthält eine Fülle von Informationen und Tipps zu http://wiki.linuxcnc.org.

Fehlerberichte

Melden Sie Fehler an den LinuxCNC GitHub bug tracker.

1.2. System-Anforderungen

1.2.1. Mindestanforderungen

Das minimale System, um LinuxCNC unter Debian/Ubuntu zu nutzen variiert mit der jeweiligen Anwendung. Stepper-Systeme benötigen im Allgemeinen schnellere Threads, um Schrittimpulse zu erzeugen, als Servo-Systeme. Sie können die Live-CD verwenden, um die Software zu testen, bevor Sie sich für eine permanente Installation auf einem Computer entscheiden. Beachten Sie, dass die Zahlen des Latenz-Tests für die Software-Schritterzeugung wichtiger sind als die Prozessorgeschwindigkeit. Mehr Informationen über den Latency Test sind hier. Außerdem muss LinuxCNC auf einem Betriebssystem ausgeführt werden, das einen speziell modifizierten Kernel verwendet, siehe Kernel and Version Requirements.

Weitere Informationen finden Sie auf der LinuxCNC Wiki Seite: Hardware Requirements

LinuxCNC und Debian Linux sollte einigermaßen gut auf einem Computer mit den folgenden minimalen Hardware-Spezifikationen laufen. Diese Zahlen sind nicht das absolute Minimum, sondern wird eine angemessene Leistung für die meisten Stepper-Systeme geben.

- 1,2 GHz 64-bin x86-Prozessor oder Raspberry Pi 4 oder besser.
- 512 MB RAM, 4 GB mit GUI, um Überraschungen zu vermeiden

- Keine Festplatte für Live CD, 8 GB oder mehr für dauerhafte Installation
- Grafikkarte mit einer Auflösung von mindestens 1024x768, die nicht die proprietären NVidia- oder ATI-Treiber verwendet. Moderne Onboard-Grafikchipsätze scheinen im Allgemeinen in Ordnung zu sein.
- Internetverbindung (nicht unbedingt erforderlich, aber sehr nützlich für Updates und für die Kommunikation mit der LinuxCNC-Community)

Die Mindestanforderungen an die Hardware ändern sich mit der Weiterentwicklung der Linux-Distributionen, daher sollten Sie sich auf der Debian-Website über die Details der von Ihnen verwendeten Live-CD informieren. Bei älterer Hardware kann es von Vorteil sein, eine ältere Version der Live-CD zu wählen, wenn diese verfügbar ist.

Wenn Sie nicht auf die Verteilung von direkt ausführbaren Programmen (engl. binaries) vertrauen wollen und/oder zum Quellbaum von LinuxCNC beitragen wollen, dann gibt es eine gute Chance, dass Sie einen zweiten Computer zur Ausführung der Compilation benötigen. Obwohl LinuxCNC und Ihre Entwicklungen wahrscheinlich gleichzeitig in Bezug auf Festplattenspeicher, RAM und sogar CPU-Geschwindigkeit ausgeführt werden können, wird eine Maschine, die beschäftigt ist, schlechtere Latenzzeiten haben, so dass es unwahrscheinlich ist, dass Sie Ihren LinuxCNC Quellbaum zu kompilieren und gleichzeitig Chips produzieren.

1.2.2. Kernel- und Versionsanforderungen

LinuxCNC erfordert einen Kernel, der für die Echtzeitnutzung modifiziert wurde, um echte Maschinenhardware zu steuern. Es kann jedoch auf einem Standard-Kernel im Simulationsmodus für Zwecke wie die Überprüfung G-Code, Testen von Konfigurationsdateien und Lernen des Systems laufen. Um mit diesen Kernel-Versionen arbeiten zu können, werden zwei Versionen von LinuxCNC verteilt. Die Paketnamen sind "linuxcnc" und "linuxcnc-uspace".

Die Echtzeit-Kerneloptionen sind preempt-rt, RTAI und Xenomai.

Sie können die Kernel-Version Ihres Systems mit dem folgenden Befehl ermitteln:

```
uname -a
```

Wenn Sie (wie oben) `-rt-` im Kernel-Namen sehen, dann laufen Sie mit dem preempt-rt Kernel und sollten die "uspace" Version von LinuxCNC installieren. Sie sollten auch uspace für "sim"-Konfigurationen auf Nicht-Echtzeit-Kerneln installieren.

Wenn Sie `-rtai-` im Kernel-Namen sehen, dann laufen Sie mit RTAI-Echtzeit. Siehe unten für die LinuxCNC Version zu installieren.

Preempt-RT mit dem Paket *linuxcnc-uspace*

Preempt-RT ist das jüngste der Echtzeitsysteme und ist auch die Version, die einem Mainline-Kernel am nächsten kommt. Preempt-RT-Kernel sind als vorkompilierte Pakete in den Haupt-Repositorien verfügbar. Mit dem Suchbegriff "PREEMPT_RT" werden sie gefunden, und man kann sie wie jedes andere Paket herunterladen und installieren. Preempt-RT bietet im Allgemeinen die beste

Treiberunterstützung und ist die einzige Option für Systeme, die Mesa-Ethernet Hardware-Treiberkarten verwenden. Im Allgemeinen hat preempt-rt die schlechteste Latenzzeit der verfügbaren Systeme, aber es gibt Ausnahmen.

RTAI mit *linuxcnc*-Paket

RTAI ist seit vielen Jahren die Hauptstütze der LinuxCNC-Distributionen. Es wird in der Regel die beste Echtzeit-Leistung in Bezug auf niedrige Latenz, aber möglicherweise schlechtere Peripherie-Unterstützung und nicht so viele Bildschirmauflösungen haben. Ein RTAI-Kernel ist im LinuxCNC-Paket-Repository verfügbar. Wenn Sie aus dem Live/Install-Image installiert haben, wird der Wechsel zwischen Kernel und LinuxCNC-Flavour in [Installing-RTAI] beschrieben.

Xenomai mit *linuxcnc-uspace* Paket

Xenomai wird auch unterstützt, aber Sie müssen den Kernel finden oder bauen und LinuxCNC aus den Quellen kompilieren, um es zu nutzen.

RTAI mit *linuxcnc-uspace*-Paket

Es ist auch möglich, LinuxCNC mit RTAI im User-Space-Modus zu betreiben. Wie bei Xenomai müssen Sie dazu aus dem Quellcode kompilieren.

1.2.3. Problematische Hardware

Laptops

Laptops sind im Allgemeinen nicht für die Erzeugung von Softwareschritten in Echtzeit geeignet. Auch hier wird Ihnen ein Latenztest über einen längeren Zeitraum die Informationen liefern, die Sie benötigen, um die Eignung festzustellen.

Videokarten

Wenn Ihre Installation mit einer Bildschirmauflösung von 800 x 600 erscheint, erkennt Debian höchstwahrscheinlich Ihre Grafikkarte oder Ihren Monitor nicht. Dies kann manchmal durch die Installation von Treibern oder die Erstellung/Bearbeitung von Xorg.conf-Dateien umgangen werden.

1.3.Hardware-Schnittstelle

1.3.1. Hardware zur Ausführung der LinuxCNC Software

Es wird ein Computer benötigt, um LinuxCNC auszuführen. Siehe Computer Anforderungen

Am häufigsten wird ein x86 Computer (Standard Intel / AMD Computer) verwendet

ARM-Computer wie der Raspberry Pi oder Orange Pi können verwendet werden

1.3.2. Hardware Schnittstelle zur CNC-Maschine

Eine Schnittstelle (engl. interface) ist notwendig, um Signale und Informationen zwischen LinuxCNC (die Software auf dem Computer) und CNC-Hardware (wie Stepper / Servotreiber, Endschalter, Eingänge und Ausgänge etc.) zu übertragen (und zu konvertieren). Es gibt mehrere verschiedene Möglichkeiten für die Hardware, um sich zu begegnen.

Schnittstellen können sein: - Parallelport - Ethernet - EtherCAT - PCI / PCIe - SPI (wobei der Computer eine native SPI-Schnittstelle hat, wie der Raspberry Pi) - USB (das ist _keine _ Echtzeit Schnittstelle)

Ein Mix aus verschiedenen Schnittstellen kann verwendet werden. Zum Beispiel eine Kombination von Ethercat für Servoantriebe und Parallelport für zusätzliche General Purpose Inputs / Outputs (GPIO)

Einige dieser Lösungen sind für alle Aspekte des Hardware-Interfacing nutzbar und einige haben spezifische Rollen (z.B. non-Echtzeit GPIO (engl. Abkürzung für Allzweck-IO) für eine Bediener-Schnittstelle mit Schaltern).

Optionen für Hardware-Schnittstellen ändern sich im Laufe der Zeit. Diese Liste ist keine 100% vollständige Liste aller Hardware-Schnittstellen, die mit LinuxCNC verwendet werden können.

1.3.3. Parallelport

Verwenden eines Mainboard Parallel Ports, oder eine PCI/PCIe Parallelport-Karte.

1.3.4. Parallel-Port Software Schnittstelle

Echtzeit- (zeitkritische) Aufgaben wie Schritt-Generierung werden in Software auf dem LinuxCNC-Host durchgeführt - das bedeutet, dass die parallele Port-Schnittstelle viel empfindlicher für die Latenz des LinuxCNC-Computers ist.

Vorteile: - Niedrige Kosten - Einfache Konfiguration

Nachteile: - Sensitiv für die Latenz des LinuxCNC-Computers - Begrenzte (engl. limited) Eingänge/Ausgänge - Einige PCI/PCIe parallele Portkarten funktionieren nicht gut oder unterstützen den EPP-Modus nicht richtig (EPP-Modus ist für die parallele Port-Schnittstelle zu Mesa / PICO-Karten erforderlich).

1.3.5. Parallel Port FPGA Kommunication

Echtzeit (zeitkritische) Aufgaben wie Schritt-Generierung werden in Hardware (nicht auf dem Computer) ausgeführt

Mesa über den Parallelport

Mesa nutzt Field-programmable gate array (FPGA), die über die parallele Schnittstelle (engl. parallel port) angeschlossen sind, wie zum beispiel die 7i43

Website: https://mesanet.com/ Store: https://store.mesanet.com/

Pico Systems über den Parallelport

https://pico-systems.com/univstep.html

1.3.6. Ethernet

Mesa Ethernet

Mesa-Karten mit einem feldprogrammierbaren Gate-Array (FPGA), an LinuxCNC-Computer über Ethernet angeschlossen. Zeitkritische (Realtime) Aufgaben werden auf der FPGA-Karte ausgeführt.

Mehrere Ethernet-Schnittstelle FPGA-Karten sind verfügbar, mit vielen Erweiterungskarten

Website: https://mesanet.com/ Store: https://store.mesanet.com/

Remora Ethernet

Echtzeit-Anforderungen werden auf eine Steuerungsplatine (engl. controller boards) abgegeben. Mehrere verschidene Steuerungsplatinen werden unterstützt, siehe Remora docs.

Beachten Sie, dass einige dieser Controller-Boards (z.B. NVEM, EC300, EC500) für den Einsatz mit Mach3 konzipiert/verkauft werden. Die Verwendung mit LinuxCNC erfordert das Flashen neuer Firmware, die von der LinuxCNC-Community entwickelt wurde. Der Hersteller unterstützt LinuxCNC nicht.

Expatria Technologies PicoBOB-DLX wurde entwickelt für LinuxCNC Remora.

Remora Dokumentation: https://remora-docs.readthedocs.io

As of March 2024:

```
STM32 based controller boards

    NVEM - ein STM32F207-basiertes Board mit Ethernet PHY Chip, das ursprünglich für
Mach3 bestimmt war. [Wird nicht mehr produziert, Legacy Support - keine neuen Features]

    EC500 - ein STM32F407-basiertes Board mit Ethernet PHY Chip, das ursprünglich für
Mach3 bestimmt war. [Wird nicht mehr produziert, Legacy Support - keine neuen Features]

    Expatria Technologies Flexi-HAL mit uFlexiNET Ethernet Adapter - ein STM32F446-
basiertes Board mit W5500 Ethernet SPI Adapter für Remora

iMX RT1052 based controller boards

    NVEM, EC300 & EC500 - iMXRT1052-basierte Controller-Boards mit Ethernet PHY-Chip,
ursprünglich für Mach3 bestimmt. [In aktiver Entwicklung]

RP2040 based controller boards

    WIZnet W5500-EVB-Pico - Raspberry Pi RP2040 basiertes Entwicklungsboard mit on-board
W5500 Ethernet SPI Adapter

    Expatria Technologies PicoBOB-DLX - Raspberry Pi RP2040 based board with on-board
```

```
W5500 Ethernet SPI adapter designed for Remora
```

Litex-CNC

Dieses Projekt zielt darauf ab, eine generische CNC-Firmware und Treiber für FPGA-Karten zu erstellen, die von LiteX unterstützt werden. Die Konfiguration von Board und Treiber erfolgt über json-Dateien. Die unterstützten Boards sind die Colorlight Boards 5A-75B und 5A-75E, da diese mit der Open Source Toolchain vollständig unterstützt werden.

Colorlight 5A-75B und 5A-75E-Karten sind als LED-Empfängerkarte konzipiert - mit Ausgabe auf LED-Matrix-Panels. Diese Karten haben nur Ausgänge - Hardware-Änderungen sind erforderlich, um auch Eingaben zu ermöglichen. Hierzu muss gelötet werden. Ausgangspuffer können durch einen Eingangspuffer ersetzt werden.

https://litex-cnc.readthedocs.io

LinuxCNC-RIO

RealtimeIO für LinuxCNC basierend auf einem FPGA

Ethernet-Schnittstelle kann mit einer Ethernet-SPI-Schnittstelle verwendet werden.

https://github.com/multigcs/LinuxCNC-RIO

1.3.7. EtherCAT

Beckhoff EtherCAT™ und kompatible Systeme können mit der Open Source Etherlab Software mit LinuxCNC arbeiten.

EtherCAT ist das von Beckhoff ursprünglich entwickelte offene Echtzeit-Ethernet-Netzwerk. Der EtherCAT Master (LinuxCNC Computer) verwendet eine Standard-Ethernet (Netzwerk)-Schnittstelle - keine spezielle Hardware wird auf dem Master benötigt. Die Slaves verwenden spezielle Hardware. Es gibt viele EtherCAT Slave-Geräte, einschließlich Servoantriebe, Stepper-Antriebe, Eingabe, Ausgabeschnittstellen, VFDs und andere.

https://github.com/linuxcnc-ethercat/linuxcnc-ethercat

1.3.8. PCI / PCIe

Mesa

Mesa PCI / PCIe Karten mit einem Feld-Programmierbaren Gate-Array (FPGA). Zeitkritische (Echtzeit) Aufgaben werden auf der FPGA-Karte ausgeführt.

Mehrere Tochter (engl. daughter) / Erweiterungskarten sind verfügbar

Website: https://mesanet.com/ Store: https://store.mesanet.com/

1.3.9. SPI

SPI = Serial Peripheral Interface. SPI-Schnittstellen finden Sie auf single board Computern wie Raspberry Pi oder Orange Pi. Eine SPI-Schnittstelle ist in der Regel *nicht* auf Standard-Computern (AMD/Intel) vorhanden.

Remora SPI

Echtzeit-Anforderungen werden an eine Steuerungsplatine übertragen. https://remora-docs.readthedocs.io

LinuxCNC-RIO

RealtimeIO für LinuxCNC basierend auf einem FPGA

https://github.com/multigcs/LinuxCNC-RIO

Mesa

Mesa-Karten mit einem Feld-Programmierbaren Gate-Array (FPGA), das über SPI an den LinuxCNC-Computer angeschlossen ist. Zeitkritische (Realtime) Aufgaben werden auf der FPGA-Karte ausgeführt.

Beispiel: 7C80 für Raspberry Pi

Website: https://mesanet.com/ Store: https://store.mesanet.com/

1.3.10. USB

USB-Geräte können nicht verwendet werden, um Motoren zu steuern oder andere *"Echtzeit"* (engl. realtime) Aufgaben auszuführen.

USB zu Parallel-Port-Konverter sind *NICHT* verwendbar / geeignet für den CNC-Einsatz.

LinuxCNC_ArduinoConnector

Dieses Projekt ermöglicht es Ihnen, einen Arduino mit LinuxCNC zu verbinden und bietet so viele I/Os, wie Sie es je wünschen konnten. Diese Software wird als I/O-Erweiterung für LinuxCNC verwendet. Sie ist *NICHT* gedacht für Timing und sicherheitsrelevante I/Os. Verwenden Sie diese nicht für den Notaus- oder Endschalter!

Site: https://github.com/AlexmagToast/LinuxCNC_ArduinoConnector

USB zu RS485 / Modbus

USB an serielle (RS485 / Modbus) Adapter können zur Steuerung von *nicht-Echtzeit* Hardware wie variable Frequenzantriebe (VFD) zur Spindelsteuerung verwendet werden. Nicht für zeitkritische Aufgaben geeignet.

1.4. Beziehen von LinuxCNC

Dieser Abschnitt beschreibt den empfohlenen Weg zum Herunterladen und zur Neuinstallation von LinuxCNC. Für die Abenteuerlustigen gibt es auch alternative Installationsmethoden . Wenn Sie eine bestehende Installation haben, die Sie aktualisieren möchten, gehen Sie stattdessen zum Abschnitt LinuxCNC aktualisieren.

NOTE

Um Maschinen zu bedienen, benötigt LinuxCNC einen speziellen Kernel mit Echtzeit-Erweiterungen. Hier gibt es drei Möglichkeiten: preempt-rt, RTAI oder Xenomai. Darüber hinaus gibt es zwei Versionen von LinuxCNC, die mit diesen Kerneln arbeiten. Siehe die nachfolgende Tabelle für Details. Für Code-Tests und Simulationen ist es jedoch möglich, die `linuxcnc-uspace`-Anwendung auf einem regulären Kernel der Distribution zu betreiben.

Neue Installationen von LinuxCNC werden am einfachsten mit dem Live/Install Image erstellt. Dies ist ein hybrides ISO-Dateisystem-Image, das auf ein USB-Speichergerät oder eine DVD geschrieben und zum Booten eines Computers verwendet werden kann. Beim Booten haben Sie die Wahl, das "Live"-System zu starten (um LinuxCNC auszuführen, ohne irgendwelche dauerhaften Änderungen an Ihrem Computer vorzunehmen) oder den Installer zu starten (um LinuxCNC und sein Betriebssystem auf der Festplatte Ihres Computers zu installieren).

Der Prozess sieht grob umrissen wie folgt aus:

1. Laden Sie das Live/Installations-Image herunter.
2. Schreiben Sie das Image auf ein USB-Speichergerät oder eine DVD.
3. Booten Sie das Live-System, um LinuxCNC zu testen.
4. Booten Sie das Installationsprogramm, um LinuxCNC zu installieren.

1.4.1. Das Festplattenabbild (Imagedatei) herunterladen

In diesem Abschnitt werden einige Möglichkeiten zum Herunterladen des Live/Install Image beschrieben.

Normales Herunterladen

Software für LinuxCNC zum Download wird auf der Website des Projekts Downloads-Seite vorgestellt. Die meisten Benutzer werden für das Festplattenbild für Intel/AMD-PCs zielen, die URL ähnelt https://www.linuxcnc.org/iso/linuxcnc_2.9.8-amd64.hybrid.iso.

Für den Raspberry Pi werden mehrere Bilder bereitgestellt, um Unterschieden zwischen RPi4 und RPi5 gerecht zu werden.

NOTE

Verwenden Sie nicht die regelmäßige Raspbian-Distribution für LinuxCNC, die Sie möglicherweise zusammen mit Ihrem RPi Starter-Kit erhalten haben - das wird nicht den Echtzeit-Kernel nutzen und Sie können nicht von Raspbian auf Debians Kernel migrieren.

Herunterladen mit zsync

zsync ist eine Download-Anwendung, die unterbrochene Downloads wieder aufnimmt und große Dateien mit kleinen Änderungen effizient überträgt (wenn Sie eine ältere lokale Kopie haben). Bitte beachten Sie, dass es das http Protokoll nutzt, nicht https. Verwenden Sie zsync, wenn das Herunterladen des Abbildes mit der Methode Normales Herunterladen häufig unterbrochen wird.

zsync unter Linux

1. Installieren Sie zsync mit Synaptic oder indem Sie Folgendes in einem Terminal-Programm ausführen

```
sudo apt-get install zsync
```

2. Führen Sie dann diesen Befehl aus, um das ISO-Abbild auf Ihren Computer herunterzuladen

```
zsync https://www.linuxcnc.org/iso/linuxcnc_2.9.8-amd64.hybrid.iso
```

Bitte denken Sie daran, den Hash-Wert (checksum) der heruntergeladenen .ISO-Datei wie nachfolgend beschrieben zu bestätigen, da die Authentizität des Servers mit dem HTTP-Protokoll nicht garantiert ist.

zsync unter Windows

Es gibt eine Windows-Portierung von zsync. Sie funktioniert als Konsolenanwendung und kann von https://www.assembla.com/spaces/zsync-windows/documents heruntergeladen werden.

Überprüfen des Abbilds

(Dieser Schritt ist nicht erforderlich, wenn Sie zsync verwendet haben)

1. Überprüfen Sie nach dem Herunterladen die Prüfsumme des Abbildes, um die Integrität sicherzustellen.

```
md5sum linuxcnc-2.9.8-amd64.iso
```

oder

```
sha256sum linuxcnc-2.9.8-amd64.iso
```

1. Vergleichen Sie dann mit diesen Prüfsummen

```
amd64 (PC)
md5sum: cf77d61fcba9641d7205ac33751e5f38
sha256sum: 72eab92d7c34c238b0429054dc52d240df8dc5f083e769a39194cfac3e4984e8
arm64 (Pi)
md5sum: 4547e8a72433efb033f0a5cf166a5cd2
sha256sum: ff3ba9b8dfb93baf1e2232746655f8521a606bc0fab91bffc04ba74cc3be6bf0
```

Überprüfen der md5 Prüfsumme unter Windows oder Mac

Windows wird nicht mit einem md5sum-Programm ausgeliefert, aber es gibt Alternativen. Weitere

Informationen finden Sie unter: How To MD5SUM

1.4.2. Das Abbild auf ein bootfähiges Gerät schreiben

Das LinuxCNC Live/Install-ISO-Image ist ein hybrides ISO-Image, das direkt auf ein USB-Speichergerät (USB-Stick) oder eine DVD geschrieben und zum Booten eines Computers verwendet werden kann. Das Image ist zu groß, um auf eine CD zu passen.

Raspberry Pi Image

Das Raspberry Pi-Image ist ein komplettes SD-Karten-Image und sollte mit der [Raspberry Pi Imager App](https://www.raspberrypi.com/software/) auf eine SD-Karte geschrieben werden. Die imager app kann die .zip Datei direkt öffnen, sie muss also nicht entpackt werden.

AMD-64 (x86-64, PC) Festplatten-Image mit GUI-Tools

Downloaden und installieren Sie Balena Etcher (Linux, Windows, Mac) und schreiben Sie das heruntergeladene Bild auf ein USB-Laufwerk.

Wenn Ihr Bild nicht bootet, versuchen Sie bitte auch Rufus. Es sieht komplizierter aus, scheint aber mit verschiedenen BIOSen kompatibel zu sein.

Linux Kommandozeile (engl. command line)

1. Schließen Sie ein USB-Speichergerät an (z. B. ein Flash-Laufwerk oder ein Gerät vom Typ Thumb Drive).
2. Ermitteln Sie die Gerätedatei, die dem USB-Flash-Laufwerk entspricht. Diese Information finden Sie in der Ausgabe von `sudo dmesg`, nachdem Sie das Gerät angeschlossen haben. `cat /proc/partitions` kann ebenfalls hilfreich sein.
3. Verwenden Sie den Befehl `dd`, um das Image auf Ihr USB-Speichergerät zu schreiben. Wenn Ihr Speichergerät zum Beispiel als `/dev/sde` angezeigt wird, verwenden Sie diesen Befehl:

```
dd if=linuxcnc_2.9.8-amd64.hybrid.iso of=/dev/sde bs=4k status=progress
```

Kommandozeile- MacOS

1. Öffnen Sie ein Terminal und geben Sie ein

```
diskutil list
```

2. Stecken Sie den USB-Stick ein und notieren Sie sich den Namen der neuen Festplatte, die angezeigt wird, z. B. /dev/disk5.
3. Trennen Sie den USB-Anschluss. Die oben gefundene Zahl sollte anstelle des N ersetzt werden.

```
diskutil unmountDisk /dev/diskN
```

4. Übertragen Sie die Daten mit dd, wie oben für Linux beschrieben. Beachten Sie, dass der Datenträgername ein "r" am Anfang hat.

```
sudo dd if=linuxcnc_2.9.8-amd64.hybrid.iso of=/dev/rdiskN bs=1m status=progress
```

Schreiben des Abbilds auf eine DVD unter Linux

1. Legen Sie einen DVD-Rohling in Ihren Brenner ein. Ein Fenster "CD/DVD Creator" oder "Disc-Typ auswählen" wird angezeigt. Schließen Sie es, da wir es nicht verwenden werden.
2. Suchen Sie das heruntergeladene Bild im Dateibrowser.
3. Klicken Sie mit der rechten Maustaste auf die ISO-Image-Datei und wählen Sie Write to Disc.
4. Wählen Sie die Schreibgeschwindigkeit. Es wird empfohlen, mit der niedrigstmöglichen Geschwindigkeit zu schreiben.
5. Starten Sie den Brennvorgang.
6. Wenn ein Fenster "Wählen Sie einen Dateinamen für das Disk-Image" erscheint, wählen Sie einfach OK.

Schreiben des Abbilds auf eine DVD unter Windows

1. Downloaden und installieren Sie Infra Recorder, ein kostenloses und quelloffenes Programm zum Brennen von Images (slang für Festplattenabbilder), herunter und installieren Sie es: https://infrarecorder.org/.
2. Legen Sie eine leere CD in das Laufwerk ein und wählen Sie Nichts tun oder Abbrechen, wenn ein Dialogfeld für die automatische Ausführung erscheint.
3. Öffnen Sie Infra Recorder, wählen Sie das *Actions* Menü, dann *Burn image*.

Schreiben des Abbilds auf eine DVD unter Mac OSX

1. Die .iso-Datei herunterladen
2. Klicken Sie mit der rechten Maustaste auf die Datei im Finder-Fenster und wählen Sie "Auf einen Datenträger brennen". (Die Option zum Brennen auf einen Datenträger wird nur angezeigt, wenn der Computer über ein optisches Laufwerk verfügt oder angeschlossen ist.)

1.4.3. LinuxCNC ausprobieren

Schalten Sie den Computer mit dem angeschlossenen USB-Speichergerät oder der DVD im DVD-Laufwerk aus und schalten Sie ihn dann wieder ein. Dadurch wird der Computer vom Live/Installationsabbild gebootet und die Option Live-Boot gewählt.

NOTE Wenn das System nicht von der DVD oder dem USB-Stick bootet, kann es erforderlich sein, die Bootreihenfolge im PC-BIOS zu ändern.

Sobald der Computer hochgefahren ist, können Sie LinuxCNC ausprobieren, ohne es zu installieren. Sie können keine benutzerdefinierten Konfigurationen erstellen oder ändern die meisten Systemeinstellungen in einer Live-Sitzung, aber Sie können (und sollten) den Latenz-Test durchführen.

Um LinuxCNC auszuprobieren: Wählen Sie aus dem Menü Anwendungen/CNC den Eintrag LinuxCNC. Es

öffnet sich ein Dialogfeld, aus dem Sie eine von vielen Beispielkonfigurationen auswählen können. An diesem Punkt ist es nur wirklich sinnvoll, eine "sim" Konfiguration zu wählen. Einige der Beispielkonfigurationen enthalten auf dem Bildschirm 3D simulierte Maschinen, suchen Sie nach "Vismach", um diese zu sehen.

Um festzustellen, ob Ihr Computer für die Erzeugung von Software-Schrittimpulsen geeignet ist, führen Sie den Latenztest wie folgt aus: here.

Zum Zeitpunkt des Schreibens des Live-Image ist nur mit dem preempt-rt Kernel und einem passenden LinuxCNC verfügbar. Auf mancher Hardware bietet dies möglicherweise keine ausreichende Latenzzeit. Es gibt eine experimentelle Version, die den RTAI-Echtzeit-Kernel verwendet, der oft eine bessere Latenzzeit bietet.

1.4.4. LinuxCNC installieren

Um LinuxCNC von der Live-CD zu installieren, wählen Sie beim Booten "Install (Graphical)".

1.4.5. Updates von LinuxCNC

Mit der normalen Installation der Update-Manager wird Sie über Updates zu LinuxCNC, wenn Sie online gehen und ermöglichen es Ihnen, einfach zu aktualisieren, ohne Linux Kenntnisse erforderlich. Es ist OK, alles außer dem Betriebssystem zu aktualisieren, wenn gefragt.

WARNING	Aktualisieren Sie das Betriebssystem nicht zu einer neuen Version, wenn Sie dazu aufgefordert werden. Sie sollten jedoch Betriebssystem *Aktualisierungen* akzeptieren, insbesondere Sicherheitsaktualisierungen.

1.4.6. Probleme bei der Installation

In seltenen Fällen kann es vorkommen, dass Sie das BIOS auf die Standardeinstellungen zurücksetzen müssen, wenn während der Live-CD-Installation die Festplatte beim Booten nicht erkannt wird.

1.4.7. Alternative Methoden für eine Installation

4Der einfachste und bevorzugte Weg, LinuxCNC zu installieren, ist die Verwendung des Live/Install Image wie oben beschrieben. Diese Methode ist so einfach und zuverlässig wie wir sie machen können und eignet sich für Anfänger und erfahrene Benutzer gleichermaßen. Jedoch wird in der Regel diese alle vorhandenen Betriebssysteme auf Ihrer Festplatte überschreiben. Wenn Sie Dateien auf Ihrem Ziel-PC haben, die Sie noch behalten möchten, so wählen eine der in diesem Abschnitt beschriebenen Methoden.

Zusätzlich, für erfahrene Benutzer, die mit der Debian-Systemadministration vertraut sind (Finden von Installations-Images, Manipulieren von apt-Quellen, Ändern von Kernel-Flavors usw.), werden neue Installationen auf den folgenden Plattformen unterstützt: ("amd64" bedeutet "64-Bit" und ist nicht spezifisch für AMD-Prozessoren, es läuft auf jedem 64-Bit-x86-System)

Distribution	Architektur	Kernel	Paket-Name	Typische Verwendung
Debian Trixie	amd64 & armhf	preempt-rt	linuxcnc-uspace	Maschinensteuerung und -simulation
Debian Trixie	amd64	RTAI	linuxcnc	Maschinensteuerung (engl. machine control)
Debian Bookworm	amd64 & armhf	preempt-rt	linuxcnc-uspace	Maschinensteuerung und -simulation
Debian Bookworm	amd64	RTAI	linuxcnc	Maschinensteuerung (engl. machine control)
Debian Bullseye	amd64	preempt-rt	linuxcnc-uspace	Maschinensteuerung und -simulation
Jede (engl. any)	Jede (engl. any)	LinuxCNC direkt nach der Installation	linuxcnc-uspace	NUR Simulation

NOTE LinuxCNC v2.9 ist nicht lauffähig auf Debian 9 und früheren Debian Versionen.

Preempt-RT-Kernel

Die Preempt-rt-Kernel sind für Debian aus dem regulären debian.org-Archiv verfügbar. Das Paket heißt `linux-image-rt-*` Installieren Sie das Paket einfach wie jedes andere Paket über den Synaptic-Paketmanager oder mit apt-get über die Kommandozeile.

RTAI-Kernel

Die RTAI-Kernel stehen im linuxcnc.org-Debian-Archiv zum Download bereit. Die apt-Quelle ist:

- Debian Trixie: `deb http://linuxcnc.org trixie base`
- Debian Bookworm: `deb http://linuxcnc.org bookworm base`
- Debian Bullseye: `deb http://linuxcnc.org bullseye base`
- Debian Buster: `deb http://linuxcnc.org buster base`

LinuxCNC und der RTAI kernel sind jetzt nur für 64-bit Betriebssysteme verfügbar, aber es gibt nur sehr wenige überlebende Systeme, die nicht mit einem 64-Bit-Betriebssystem arbeiten können.

Installation auf Debian Trixie (mit Preempt-RT-Kernel)

1. Installieren Sie Debian Trixie (Debian 13), Version amd64. Sie können das Installationsprogramm hier herunterladen: https://www.debian.org/distrib/
2. Wenn Sie nach dem Brennen der Iso und dem Booten den Gnome-Desktop nicht möchten, wählen Sie "Erweiterte Optionen" > "Alternative Desktop-Umgebungen" und wählen Sie die gewünschte aus.

Dann wählen Sie "Installation" oder "Graphische Installation".

WARNING	Geben Sie kein root-Passwort ein, da sonst sudo deaktiviert ist und Sie die folgenden Schritte nicht abschließen können.

3. Führen Sie das Folgende in einem Terminal aus, um den Rechner auf den neuesten Stand der Pakete zu bringen.

```
sudo apt-get update
sudo apt-get dist-upgrade
```

NOTE	It is possible to download a version of LinuxCNC directly from Debian (currently version 2.9.4) but a more up-to-date version (2.9.8) can be installed from the LinuxCNC repository.

4. Installieren Sie den Preempt-RT kernel and Module

```
sudo apt-get install linux-image-rt-amd64
```

5. Starten Sie neu und wählen Sie den Linux 6.1.0-10-rt-amd64 Kernel. Die genaue Kernel-Version kann abweichen, suchen Sie nach dem Suffix "-rt". Dies könnte im Untermenü "Erweiterte Optionen für Debian Bookworm" in Grub versteckt sein. Wenn Sie sich anmelden, stellen Sie sicher, dass *PREEMPT RT* mit dem folgenden Befehl gemeldet wird.

```
uname -v
```

6. Öffnen Sie Menü Anwendungen > System > Synaptic Package Manager, suchen Sie nach *linux-image*, klicken mit der rechten Maustaste auf das ursprüngliche Nicht-rt und wählen Sie "Zur vollständigen Entfernung markieren". Neu starten. Damit wird das System gezwungen, vom RT-Kernel zu booten. Wenn Sie es vorziehen, beide Kernel beizubehalten, müssen die anderen Kernel nicht gelöscht werden, aber es sind Änderungen an der Grub-Boot-Konfiguration erforderlich, die den Rahmen dieses Dokuments sprengen.
7. Fügen Sie den LinuxCNC Archive Signing Key zu Ihrem apt keyring hinzu, indem Sie [das LinuxCNC Installationsskript] herunterladen(https://www.linuxcnc.org/linuxcnc-install.sh). Sie müssen das Skript ausführbar machen, um es auszuführen:

```
chmod +x linuxcnc-install.sh
```

Dann können Sie den Installer ausführen:

```
sudo ./linuxcnc-install.sh
```

Installation unter Debian Trixie (mit experimentellem RTAI-Kernel)

1. Dieser Kernel und die LinuxCNC-Version können auf der Live DVD-Installation installiert werden, oder alternativ auf einer neuen Installation von Debian Trixie 64-bit, wie oben beschrieben.

2. Sie können die unterzeichnenden Schlüssel- und Repository-Informationen des LinuxCNC-Archivs hinzufügen, indem Sie das Installationsskript wie oben beschrieben herunterladen und ausführen. Wenn ein RTAI-Kernel erkannt wird, wird es vor der Installation von Paketen stoppen.
3. Aktualisieren Sie die Paketliste von linuxcnc.org

```
sudo apt-get update
```

4. Entfernen einer bereits existierenden uspace-Version von LinuxCNC und Installation eines neuen Echtzeit-Kernels, RTAI und die RTAI-Version von LinuxCNC.

```
sudo apt-get purge linuxcnc-uspace
sudo apt-get purge linuxcnc-doc*
sudo apt-get install linuxcnc
```

   Starten Sie den Rechner neu und stellen Sie sicher, dass das System mit dem neuen Kernel 5.4.258-rtai bootet.

Installieren auf Raspbian 12

Tuen Sie das nicht. Die Latenzen sind zu schlecht mit dem Standard-Kernel und der PREEMPT_RT (die RT ist wichtig) Kernel von Debian bootet nicht auf der Pi (Stand: 1/2024). Bitte beachten Sie die online zur Verfügung gestellten .iso Images auf der regulären LinuCNC download Seite. Sie können sie selbst gemäß den unter online verfügbaren Skripten erstellen.

1.5. LinuxCNC ausführen

1.5.1. Aufrufen von LinuxCNC

Nach der Installation startet LinuxCNC wie jedes andere Linux-Programm: Führen Sie es aus dem terminal aus, indem Sie den Befehl *linuxcnc* eingeben, oder wählen Sie es im Menü *Anwendungen → CNC* aus.

1.5.2. Konfiguration des Launchers (Startprogramms)

Beim Starten von LinuxCNC (aus dem CNC-Menü oder von der Kommandozeile ohne Angabe einer INI-Datei) startet der Dialog Kofigurations-Auswahl.

Im Dialogfeld "Konfigurationsauswahl" kann der Benutzer eine seiner vorhandenen Konfigurationen (Meine Konfigurationen) oder eine neue Konfiguration (aus den Beispielkonfigurationen) auswählen, die in sein Home-Verzeichnis kopiert werden soll. Die kopierten Konfigurationen werden beim nächsten Aufruf der Konfigurationsauswahl unter Meine Konfigurationen angezeigt.

Der Konfigurations Selector bietet eine Auswahl an Konfigurationen:

- *Meine Konfigurationen* - Benutzerkonfigurationen in linuxcnc/configs in Ihrem Home-Verzeichnis.
- *Beispielkonfigurationen* - Beispielkonfigurationen werden, wenn ausgewählt, nach linuxcnc/configs

kopiert. Sobald eine Beispielkonfiguration in Ihr lokales Verzeichnis kopiert wurde, bietet der Launcher sie als „Meine Konfigurationen“ an. Die Namen, unter denen diese lokalen Konfigurationen angezeigt werden, entsprechen den Namen der Verzeichnisse innerhalb des Verzeichnisses configs/:

- *sim* - Konfigurationen, die simulierte Hardware enthalten. Diese können zum Testen oder Lernen, wie LinuxCNC funktioniert, verwendet werden.
- by_interface' - Konfigurationen nach GUI geordnet.
- *by_machine* - Konfigurationen organisiert nach Maschine.
- *apps* - Anwendungen, die kein Starten von linuxcnc erfordern, aber zum Testen oder Ausprobieren von Anwendungen wie PyVCP oder GladeVCP nützlich sein können.
- *attic* - Veraltete oder historische Konfigurationen.

Die Simulationskonfigurationen sind oft der nützlichste Ausgangspunkt für neue Benutzer und sind nach unterstützten GUIs organisiert:

- *axis* - Tastatur- und Maus-GUI
- *craftsman* - Touch Screen GUI (nicht mehr weiterentwickelt????)
- *gmoccapy* - Touchscreen-GUI
- *gscreen* - Touchscreen-GUI
- *pyvcp_demo* - Python Virtual Control Panel
- *qtaxis* - Touchscreen-GUI, AXIS-Lookalike
- *qtdragon* - Touchscreen GUI
- *qtdragon_hd* - Touchscreen-GUI, detailierte Auflösung (engl. high definition)
- *qtplasmac* - Touchscreen-GUI, für Plasma-Tische
- *qttouchy* - Touchscreen-GUI
- *tklinuxcnc* - Tastatur- und Maus-GUI (wird nicht mehr gepflegt)
- *touchy* - Touchscreen-GUI
- *woodpecker* - Touchscreen-GUI

Ein GUI-Konfigurationsverzeichnis kann Unterverzeichnisse mit Konfigurationen enthalten, die spezielle Situationen oder die Einbettung anderer Anwendungen veranschaulichen.

Die *by_interface*-Konfigurationen sind um gängige, unterstützte Schnittstellen herum organisiert:

- allgemeine Mechatronik
- mesa
- parport
- pico

Um diese Konfigurationen als Ausgangspunkt für ein System zu verwenden, kann entsprechende Hardware erforderlich sein.

Die *by_machine*-Konfigurationen sind um vollständige, bekannte Systeme herum organisiert:

- boss
- cooltool
- scortbot erIII
- sherline
- smithy
- tormach

Für die Verwendung dieser Konfigurationen kann ein komplettes System erforderlich sein.

Die *Apps-Elemente* (engl. app items) sind typischerweise entweder:

1. utilities (engl. für Hilfsprogramme), für die linuxcnc nicht gestartet werden muss
2. Demonstrationen von Anwendungen, die mit LinuxCNC verwendet werden können
   - info - erstellt eine Datei mit Systeminformationen, die für die Problemdiagnose nützlich sein können.
   - gladevcp - Beispiele für GladeVCP-Anwendungen.
   - halrun - Startet halrun in einem Terminal.
   - latency - Anwendungen zur Untersuchung der Latenz
     - latency-histogram-1 - Histogramm für einzelnen Servo-Thread
     - latency-histogram - Histogram
     - latency-test - Standard-Test zur Bestimmung der Latenz
     - latency-plot - Streifendiagramm
   - parport - Anwendungen zum Testen von parport.
   - pyvcp - Beispiele für pyvcp-Anwendungen.
   - xhc-hb04 – Anwendungen zum Testen eines drahtlosen USB-MPG xhc-hb04

NOTE Im Verzeichnis Apps werden nur Anwendungen zum Kopieren in das Benutzerverzeichnis angeboten, die vom Benutzer sinnvollerweise geändert werden.

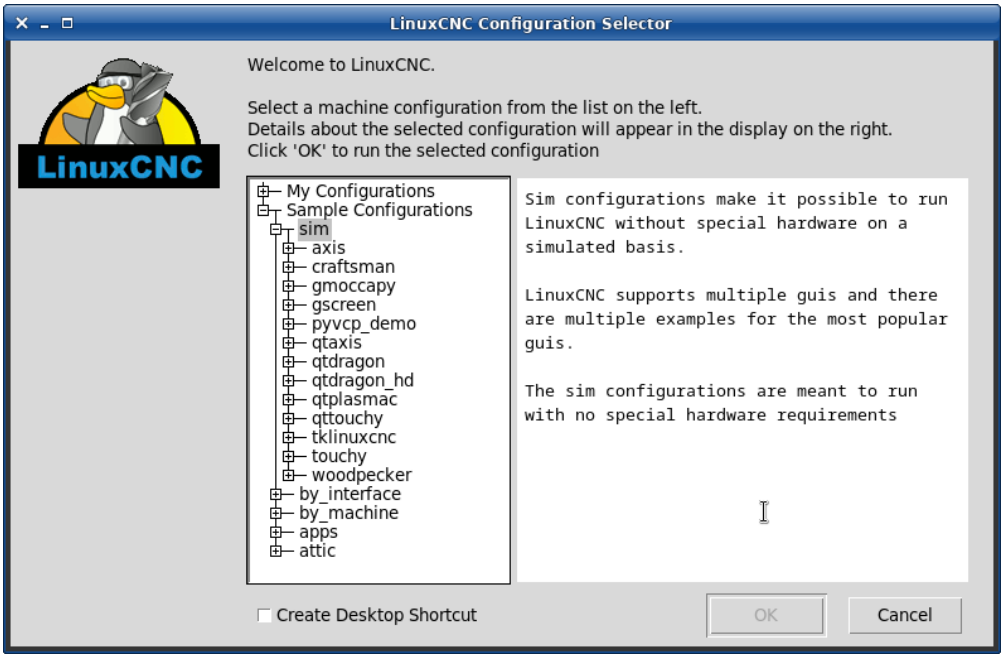


Figure 2. LinuxCNC-Konfigurationsauswahl

Klicken Sie auf eine der aufgelisteten Konfigurationen, um spezifische Informationen zu ihr anzuzeigen. Doppelklicken Sie auf eine Konfiguration oder klicken Sie auf OK, um die Konfiguration zu starten.

Wählen Sie "Desktop-Verknüpfung erstellen" und klicken Sie dann auf "OK", um ein Symbol auf dem Ubuntu-Desktop hinzuzufügen, mit dem diese Konfiguration direkt gestartet wird, ohne dass der Bildschirm "Konfigurationsauswahl" angezeigt wird.

Wenn Sie eine Konfiguration aus dem Abschnitt Beispielkonfigurationen auswählen, wird automatisch eine Kopie dieser Konfiguration im Verzeichnis ~/linuxcnc/configs abgelegt.

1.5.3. Nächste Schritte für die Konfiguration

Nachdem Sie die Beispielkonfiguration gefunden haben, die dieselbe Schnittstellenhardware wie Ihr Rechner verwendet (oder eine Simulatorkonfiguration), und eine Kopie in Ihrem Home-Verzeichnis gespeichert haben, können Sie sie an die Details Ihres Rechners anpassen. Weitere Informationen zur Konfiguration finden Sie im Integrator-Handbuch.

1.5.4. Simulator-Konfigurationen

Alle unter Beispielkonfigurationen/Sim aufgeführten Konfigurationen können auf jedem Computer ausgeführt werden. Es ist keine spezielle Hardware erforderlich und Echtzeitunterstützung ist nicht notwendig.

Diese Konfigurationen sind nützlich, um einzelne Fähigkeiten oder Optionen zu untersuchen. Die Sim-Konfigurationen sind nach der in der Demonstration verwendeten grafischen Benutzeroberfläche geordnet. Das Verzeichnis für die Achse enthält die meisten Auswahlmöglichkeiten und Unterverzeichnisse, da es sich um die am häufigsten getestete grafische Benutzeroberfläche handelt. Die Fähigkeiten, die mit einer bestimmten grafischen Benutzeroberfläche demonstriert werden, sind möglicherweise auch in anderen grafischen Benutzeroberflächen verfügbar.

1.5.5. Konfigurationsressourcen

Die Konfigurationsauswahl kopiert alle für eine Konfiguration benötigten Dateien in ein neues Unterverzeichnis von ~/linuxcnc/configs (äquivalent: /home/username/linuxcnc/configs). Jedes erstellte Verzeichnis enthält mindestens eine INI-Datei (iniflename.ini), die zur Beschreibung einer bestimmten Konfiguration verwendet wird.

When the copy happens

For the following cases the Configuration Selector copies the chosen sample configuration to ~/linuxcnc/configs:

- **Package install (deb, rpm, distro packages)**: sample configurations live under a system path and are not writabe.
- **The file is not in the directory tree of `CONFIG_DIR`** (environmental variable)
- **The file is in a directory included in `LINUXCNC_AUX_EXAMPLES`** (environmental variable)

For **Run-In-Place (RIP) builds** the source tree is normally writable by the user who built it, so the selector runs the configuration directly from the source tree without copying. Edits made through the configuration apply to the files in the RIP tree.

Forcing a copy from a RIP build

To test the copy-and-run path from a RIP build (or to keep a personal copy of a sample configuration outside the source tree), set the `debug_pickconfig` environment variable before launching LinuxCNC:

```
debug_pickconfig=1 linuxcnc
```

With this set, the selector copies the chosen sample configuration to ~/linuxcnc/configs even though the RIP source tree is writable.

Zu den Dateiressourcen innerhalb des kopierten Verzeichnisses gehören in der Regel eine oder mehrere INI-Dateien (Dateiname.ini) für zugehörige Konfigurationen und eine Werkzeugtabellendatei (Toolfilename.tbl). Darüber hinaus können die Ressourcen HAL-Dateien (Dateiname.hal, Dateiname.tcl), eine README-Datei zur Beschreibung des Verzeichnisses und konfigurationsbezogene Informationen in einer nach einer bestimmten Konfiguration benannten Textdatei (inifilename.txt) enthalten. Die beiden letztgenannten Dateien werden angezeigt, wenn Sie die Konfigurationsauswahl verwenden.

Die mitgelieferten Beispielkonfigurationen können den Parameter HALFILE (Dateiname.hal) in der Konfigurations-INI-Datei angeben, die im kopierten Verzeichnis nicht vorhanden sind, da sie sich in der HAL-Dateibibliothek des Systems befinden. Diese Dateien können in das Benutzerkonfigurationsverzeichnis kopiert und nach Bedarf vom Benutzer für Modifikationen oder Tests geändert werden. Da das Benutzerkonfigurationsverzeichnis bei der Suche nach HAL-Dateien zuerst durchsucht wird, haben lokale Änderungen dann Vorrang.

Der Konfigurationsselektor erstellt einen symbolischen Link im Benutzerkonfigurationsverzeichnis (namens hallib), der auf die System-HAL-Datei-Bibliothek verweist. Diese Verknüpfung vereinfacht das Kopieren einer Bibliotheksdatei. Zum Beispiel, um die Bibliotheksdatei core_sim.hal zu kopieren, um

lokale Änderungen vorzunehmen:

```
cd ~/linuxcnc/configs/name_of_configuration
cp hallib/core_sim.hal core_sim.hal
```

1.6. LinuxCNC updaten

Die Aktualisierung von LinuxCNC auf eine neue Nebenversion (d.h. auf eine neue Version in der gleichen stabilen Serie, z.B. von 2.9.7 auf 2.9.8) ist ein automatischer Prozess, wenn Ihr PC mit dem Internet verbunden ist. Nach einem Minor-Release wird eine Update-Aufforderung zusammen mit anderen Software-Updates angezeigt. Wenn Ihr PC nicht mit dem Internet verbunden ist, lesen Sie bitte Updating without Network.

1.6.1. Upgrade auf die neue Version

Dieser Abschnitt beschreibt, wie Sie LinuxCNC von Version 2.8.x auf eine Version 2.9.y aktualisieren. Es wird davon ausgegangen, dass Sie eine bestehende 2.8 Installation haben, die Sie aktualisieren möchten.

Um LinuxCNC von einer Version älter als 2.8 zu aktualisieren, müssen Sie zuerst upgrade your old install to 2.8, dann folgen Sie diesen Anweisungen, um auf die neue Version zu aktualisieren.

Wenn Sie keine alte Version von LinuxCNC zu aktualisieren haben, dann sind Sie am besten aus machen eine frische Installation der neuen Version, wie im Abschnitt LinuxCNC erhalten beschrieben.

Darüber hinaus ist es unter Ubuntu Precise, Debian Wheezy oder Debian Buster eine Überlegung wert, ein Backup des "linuxcnc"-Verzeichnisses auf einem Wechselmedium vorzunehmen und eine Neuinstallaion des neuesn Betriebsystems und der LinuxCNC version durchzuführen, da diese Versionen des OS 2017, 2018 bzw. 2022 ausliefen. Wenn Sie Ubuntu Lucid nutzen, dann werden Sie dies tun müssen, da Lucid nicht mehr von LinuxCNC unterstützt wird (es war EOL im Jahr 2013).

Um größere Versionen wie 2.8 auf 2.9 zu aktualisieren, wenn Sie eine Netzwerkverbindung an der Maschine haben, müssen Sie die alten linuxcnc.org Quellen des Paketmanagers *apt* in der Datei /etc/apt/sources.list deaktivieren und fügen Sie eine neue linuxcnc.org apt Quelle für 2.9 hinzu, um dann mit apt die LinuxCNC-Installation zu aktualisieren.

Die Details hängen von der Plattform ab, auf der Sie arbeiten. Öffnen Sie ein terminal und geben Sie `lsb_release -ic` ein, um diese Informationen herauszufinden:

```
lsb_release -ic
Distributor ID: Debian
Codename:       Trixie
```

Sie sollten auf Debian Bullseye, Bookworm oder Trixie oder Ubuntu 20.04 "Focal Fossa" oder neuer nutzen. LinuxCNC 2.9.x wird auf älteren Distributionen als diese nicht laufen.

Sie müssen auch prüfen, welcher Echtzeit-Kernel verwendet wird:

```
uname -r
```

```
6.1.0-10-rt-amd64
```

Wenn Sie (wie oben) `-rt-` im Kernel-Namen sehen, dann laufen Sie mit dem preempt-rt Kernel und sollten die "uspace" Version von LinuxCNC installieren. Sie sollten auch uspace für "sim"-Konfigurationen auf Nicht-Echtzeit-Kerneln installieren.

Wenn Sie `-rtai-` im Kernel-Namen sehen, dann laufen Sie mit RTAI-Echtzeit. Siehe unten für die LinuxCNC Version zu installieren. RTAI Pakete sind derzeit für Bookworm und Buster verfügbar.

Apt Sources Konfiguration

- Öffnen Sie das Fenster `Software-Quellen`. Die Vorgehensweise ist auf den drei unterstützten Plattformen leicht unterschiedlich:
  - Debian:
    - Klicken Sie auf `Anwendungsmenü`, dann `System`, dann `Synaptic Package Manager`.
    - Klicken Sie in Synaptic auf das Menü `Einstellungen` und dann auf `Repositories`, um das Fenster `Softwarequellen` zu öffnen.
  - Ubuntu Precise:
    - Klicken Sie auf das Symbol "Dash Home" oben links.
    - Geben Sie in das Feld "Suche" den Begriff "Software" ein und klicken Sie dann auf das Symbol "Ubuntu Software Center".
    - Klicken Sie im Ubuntu Software Center-Fenster auf das Menü "Bearbeiten" und dann auf "Softwarequellen...", um das Fenster "Softwarequellen" zu öffnen.
  - Ubuntu Lucid:
    - Klicken Sie auf das Menü "System", dann auf "Verwaltung" und dann auf "Synaptic Package Manager".
    - Klicken Sie in Synaptic auf das Menü `Einstellungen` und dann auf `Repositories`, um das Fenster `Softwarequellen` zu öffnen.
- Wählen Sie im Fenster "Software-Quellen" die Registerkarte "Andere Software".
- Löschen oder deaktivieren Sie alle alten linuxcnc.org-Einträge (lassen Sie alle nicht-linuxcnc.org-Zeilen unverändert).
- Klicken Sie auf die Schaltfläche "Hinzufügen" und fügen Sie eine neue apt-Zeile hinzu. Die Zeile wird auf den verschiedenen Plattformen etwas anders aussehen:

Table 1. Eine tabuläre Übersicht über Varianten des Betriebssystems und die entsprechende Konfiguration des Projektarchivs. Die Konfiguration kann in der GUI des Paketmanagers oder in der Datei /etc/apt/sources.list spezifiziert werden.

Betriebssystem / Echtzeitversion	Repository
Debian Bullseye - preempt	`deb https://linuxcnc.org bullseye base 2.9-uspace`
Debian Bookworm - preempt	`deb https://linuxcnc.org bookworm base 2.9-uspace`

Betriebssystem / Echtzeitversion	Repository
Debian Bookworm - RTAI	`deb https://linuxcnc.org bookworm base 2.9-rt`
Debian Trixie - preempt	`deb https://linuxcnc.org trixie base 2.9-uspace`
Debian Trixie - RTAI	`deb https://linuxcnc.org trixie base 2.9-rt`

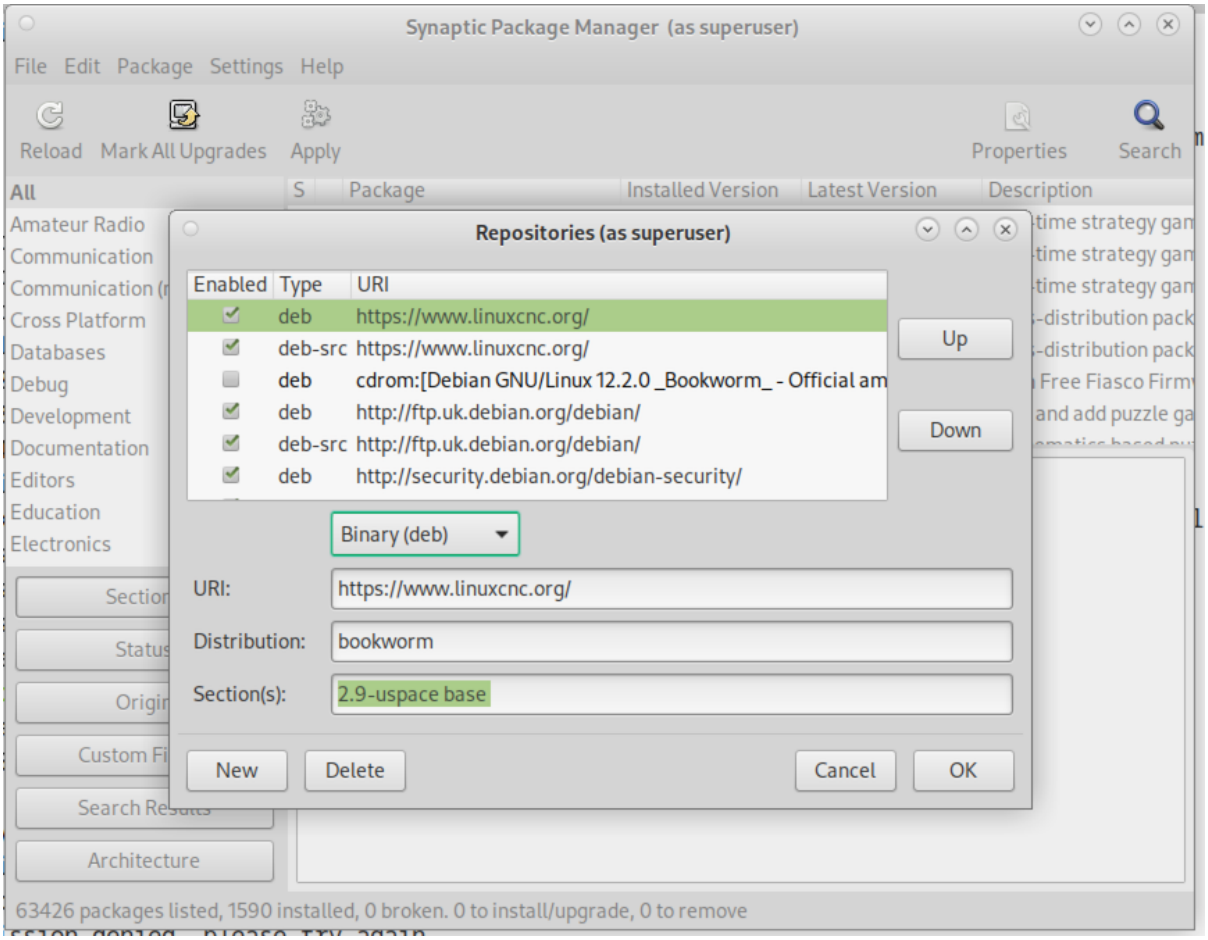


Figure 3. Abbildung mit einem Screenshot der Repository Einstellung für den Synaptic Paket Manager.

- Klicken Sie im Fenster "Softwarequellen" auf "Quelle hinzufügen" und dann auf "Schließen". Wenn ein Fenster angezeigt wird, das Sie darüber informiert, dass die Informationen über die verfügbare Software veraltet sind, klicken Sie auf die Schaltfläche "Neu laden".

Upgrade auf die neue Version

Da Ihr Computer nun weiß, wo er die neue Version der Software erhält, müssen wir sie nun installieren.

Der Prozess unterscheidet sich wiederum je nach Plattform.

Debian Bullseye, Bookworm und Trixie

Debian verwendet den Synaptic Package Manager.

- Öffnen Sie Synaptic gemäß den Anweisungen in Festlegen der apt sources oben.
- Klicken Sie auf die Schaltfläche "Neu laden".

- Verwenden Sie die Suchfunktion, um nach `linuxcnc` zu suchen.
- Das Paket heißt „linuxcnc“ für RTAI-Kernel und „linuxcnc-uspace“ für preempt-rt.
- Klicken Sie auf das Kontrollkästchen, um die neuen Pakete linuxcnc und linuxcnc-doc-* für ein Upgrade zu markieren. Der Paketmanager kann eine Reihe zusätzlicher Pakete auswählen, die installiert werden sollen, um die Abhängigkeiten zu erfüllen, die das neue linuxcnc-Paket hat.
- Klicken Sie auf die Schaltfläche "Anwenden", und lassen Sie Ihren Computer das neue Paket installieren. Das alte linuxcnc-Paket wird automatisch auf das neue Paket aktualisiert.

Ubuntu

- Klicken Sie auf das Symbol "Dash Home" oben links.
- Geben Sie in das Feld "Suche" den Begriff "Update" ein und klicken Sie dann auf das Symbol "Update Manager".
- Klicken Sie auf die Schaltfläche "Prüfen", um die Liste der verfügbaren Pakete aufzurufen.
- Klicken Sie auf die Schaltfläche "Updates installieren", um die neuen Versionen aller Pakete zu installieren.

1.6.2. Aktualisieren ohne Netzwerk

Um ohne Netzwerkverbindung zu aktualisieren, müssen Sie die .deb-Datei herunterladen und mit dpkg installieren. Die .debs können unter https://linuxcnc.org/dists/ gefunden werden.

Sie müssen im obigen Link nach unten gehen, um das richtige Debian Paket (.deb Datei) für Ihre Installation zu finden. Öffnen Sie ein Terminal und geben Sie "lsb_release -ic" ein, um den Versions-Bezeichner Ihres Betriebssystems zu finden.

```
> lsb_release -ic
Distributor ID: Debian
Codename:       trixie
```

Wählen Sie das Betriebssystem aus der Liste und dann die gewünschte Hauptversion wie 2.9-rt für RTAI oder 2.9-uspace für preempt-rt.

Wählen Sie als Nächstes den Computertyp aus, den Sie haben: binary-amd64 für 64-Bit-x86 oder binary-arm64 (64-Bit) für Raspberry Pi.

Wählen Sie dann die gewünschte Version am Ende der Liste aus, z.B. *linuxcnc-uspace_2.9.8_amd64.deb* (wählen Sie die neueste Version nach Datum). Laden Sie die deb-Datei herunter und kopieren Sie sie in Ihr Home-Verzeichnis. Sie können die Datei mit dem Dateimanager in etwas kürzeres umbenennen, wie z.B. *linuxcnc_2.9.8.deb*, dann öffnen Sie ein Terminal und installieren es mit dem Paketmanager mit diesem Befehl:

```
sudo dpkg -i linuxcnc_2.9.8.deb
```

1.6.3. Aktualisieren von Konfigurationsdateien für 2.9

Strikterer Umgang mit austauschbaren Interpretern

Wenn Sie nur normalen G-Code ausführen und nicht wissen, was ein austauschbarer Interpreter ist, wirkt sich dieser Abschnitt nicht auf Sie aus.

Ein selten genutztes Merkmal von LinuxCNC ist die Unterstützung für steckbare Interpreter, die durch die undokumentierte `[TASK]INTERPRETER` INI-Einstellung gesteuert werden.

Versionen von LinuxCNC vor 2.9.0 behandelten eine falsche *[TASK]INTERPRETER*-Einstellung, indem sie automatisch auf die Verwendung des Standard-G-Code-Interpreters zurückgriffen.

Seit 2.9.0 führt ein falscher *[TASK]INTERPRETER*-Wert dazu, dass LinuxCNC den Start verweigert. Beheben Sie diese Bedingung, indem Sie die Einstellung *[TASK]INTERPRETER* aus Ihrer INI-Datei löschen, so dass LinuxCNC den Standard-G-Code-Interpreter verwendet.

Canterp

Wenn Sie nur normalen G-Code ausführen und den austauschbaren Interpreter "canterp" nicht verwenden, hat dieser Abschnitt keine Auswirkungen auf Sie.

Für den extrem unwahrscheinlichen Fall, dass Sie `canterp` verwenden, sollten Sie wissen, dass das Modul von `/usr/lib/libcanterp.so` nach `/usr/lib/linuxcnc/canterp.so` verschoben wurde und die Einstellung `[TASK]INTERPRETER` entsprechend von `libcanterp.so` nach `canterp.so` geändert werden muss.

Spindel-Einschränkungen in der INI

Es ist nun möglich, in den [SPINDLE]-Abschnitt der INI-Datei zu konfigurieren

MAX_VELOCITY = 20000 Die maximale Spindeldrehzahl (in U/min)

MIN_FORWARD_VELOCITY = 20000 Die minimale Spindeldrehzahl (in U/min)

MAX_REVERSE_VELOCITY = 20000 Diese Einstellung wird standardmäßig auf MAX_FORWARD_VELOCITY gesetzt sofern nicht explizit angegeben.

MIN_REVERSE_VELOCITY = 3000 Diese Einstellung entspricht MIN_FORWARD_VELOCITY, jedoch für die umgekehrte Spindeldrehung. Ist dieser Wert nicht angegeben, wird sie standardmäßig auf MIN_FORWARD_VELOCITY gesetzt.

INCREMENT = 200 Legt die Schrittweite für Befehle zum Erhöhen und Verringern der Spindeldrehzahl fest. Dies kann für jede Spindel einen anderen Wert haben. Diese Einstellung ist bei AXIS und Touchy wirksam, aber beachten Sie, dass einige Kontrollbildschirme die Dinge anders handhaben können.

HOME_SEARCH_VELOCITY = 100 - Wird akzeptiert, aber ist derzeit ohne Funktion

HOME_SEQUENCE = 0 - Wird akzeptiert, ist aber derzeit ohne Funktion

1.6.4. Update von Konfigurationsdateien für 2.10.y

Touchy: Die Touchy-MACRO-Einträge gehören nun in den Abschnitt [MACROS] der INI anstelle von [TOUCHY]. Dies ist Teil der Vereinheitlichung der INI-Einstellungen zwischen den GUIs.

The mesa_modbus framework (modcompile and the mesa_modbus.c.tmpl driver template) has been removed. Configurations using it must migrate to the hm2_modbus driver with mesambccc-compiled MBCCB files, see hm2_modbus(9) and mesambccc(1). The 2.9 release retains the framework, so its documentation remains available in the 2.9 docs.

1.6.5. Preview renderer now requires OpenGL 3.3 core

The G-code preview shared by AXIS, the GTK screens (Gremlin / gmoccapy / gscreen / GladeVCP `hal_gremlin` / QtPlasmaC), and QtVCP was rewritten to use a single modern OpenGL 3.3 **core-profile** renderer (shaders, VBOs, an offscreen framebuffer for click selection, a glyph-atlas for overlay text). The legacy fixed-function path (display lists, immediate mode, `GL_SELECT` picking, `glBitmap` text, line stipple, `GL_LIGHTING`) has been removed. There is no runtime switch and no in-process fallback.

Hardware requirement. OpenGL 3.3 core is needed. On the supported platform (Linux with Mesa) this is available on Intel Sandy Bridge (2011) and newer, AMD r600 and newer, and nouveau. Machines without a capable GPU can use Mesa's software renderer (llvmpipe), which handles the line-dominated preview acceptably:

```
LIBGL_ALWAYS_SOFTWARE=1 linuxcnc myconfig.ini
```

If a core context cannot be created the GUI exits at start-up with a diagnostic naming the OpenGL 3.3 requirement and suggesting `LIBGL_ALWAYS_SOFTWARE=1`, rather than starting with a blank or corrupt preview.

WARNING

BREAKING: out-of-tree screens that inject raw legacy OpenGL

Custom screens that subclass the in-tree preview classes and **override a drawing internal to emit raw fixed-function OpenGL** (immediate mode, display lists, `glBitmap`, etc.) will fail against a core context. Compatibility is preserved only at the **calling surface**: the public methods and attributes of `rs274.glcanon.GlCanonDraw` / `glnav.GlNavBase` used by the in-tree GUIs keep their names, signatures, and behaviour (`realize`, `redraw`, `redraw_perspective`, `redraw_ortho`, `select`, `set_highlight_line`, `set_canon`, `posstrs`, the `stale_dlist(...)` cache-invalidation entry points, and the `get_*`/`is_*` callback contract). Move any custom drawing onto those supported entry points, or draw with your own modern-OpenGL code.

The immediate-mode drawing helpers of the old renderer (`linuxcnc.draw_lines`, `linuxcnc.line9`, `linuxcnc.draw_dwells`, `linuxcnc.positionlogger.call()`) are retired. They keep their names, signatures and argument checking so that out-of-tree callers still import and run, but they draw nothing and raise a `DeprecationWarning` on first use. The in-tree GUIs bake geometry to VBOs and upload the backplot from `positionlogger.points()`.

Notes for integrators and driver authors

- **AXIS / Togl.** The vendored Togl widget (`src/emc/usr_intf/axis/extensions/togl.c`) gained a boolean `-coreprofile` option (default **false**). When true it creates the context with `glXChooseFBConfig` + `glXGetVisualFromFBConfig` `glXCreateContextAttribsARB` (OpenGL 3.3 core), raising a descriptive Tcl error on failure. AXIS always enables it. The default (false) path is unchanged, so `vismach` and any out-of-tree Togl users keep their legacy contexts.
- **Gremlin (GTK).** GTK3 only hands out core contexts through `GtkGLArea`, so the gremlin widget still builds its context by hand via GLX, now requesting 3.3 core with `glXCreateContextAttribsARB`. PyOpenGL cannot resolve that extension entry point (it comes back as a null function), so gremlin loads `libGL` directly with `ctypes` and creates **and binds** the context (choose-fbconfig / make-current / swap) through that one handle to avoid mixing context pointers.
- **Line width in core profiles.** A forward-compatible core context (Qt requests one) rejects `glLineWidth(> 1)` with `GL_INVALID_VALUE` even though `GL_ALIASED_LINE_WIDTH_RANGE` reports a larger maximum. The renderer probes the accepted width once and caches it, so thick lines (the selection highlight, dwell markers) degrade to 1 px on such drivers instead of raising; a non-forward-compatible core context (AXIS's Togl, Gremlin's GLX) keeps the wider lines. Thick lines via quad expansion are a possible future improvement.
- **vismach** is unaffected: it keeps its legacy Togl context and immediate-mode drawing; only its camera consumes the (now GL-free) explicit matrices from `glnav`.

The program is built in C++ during the parse

`gcode.parse` no longer drives the preview through per-move Python callbacks. For a canon that subclasses `gcode.RendererCanon` - `rs274.glcanon.GLCanon` is one, so every in-tree preview is - the whole program is built in C++ (`GCodeRenderer`, `src/emc/rs274ngc/gcode_renderer.{hh,cc}`): the g92/rotation/g5x transform, arc segmentation, rigid taps, `(AXIS,hide)` suppression, the vertices per drawn plane, the extents, the path lengths and the dwell and tool-change records. The finished program is handed over once, at the end of the parse, as a `gcode.PreviewGeometry` through the canon's `adopt_geometry()`. A parse reads two more things off such a canon: `program_geometry` (the GEOMETRY strings and the rotation offsets) and `arcdivision`, which defaults to 64 and is read once at parse start. Every parse starts from a zero transform with nothing drawn; where the machine stands arrives as the caller's initcode (a `G53 G0` per axis), which the leading-traverse drop repositions on rather than draws. A `RendererCanon` subclass without a callable `adopt_geometry` is a `TypeError` from `gcode.parse`, not a silent fall back to callbacks.

The per-event canon protocol is unchanged for every other canon: `rs274.interpret.PrintCanon`, the interpreter tests and out-of-tree users of `gcode.parse` still receive `straight_feed`, `arc_feed`, `next_line` and the rest exactly as before, and `rs274.interpret.Translated` / `ArcsToSegmentsMixin` remain for them. The `gcode` module itself was rewritten on pybind11; its functions keep their names and signatures. One behaviour change: `gcode.linecode()` snapshots the running parse, and raises `ValueError` when no parse is in progress.

WARNING	*BREAKING: out-of-tree canons that subclass* `rs274.glcanon.GLCanon` On a rendered parse the interpreter forwards only `next_line` (on the handful of

lines that still forward, not once per line), `comment`, `message`, `change_tool`, `check_abort`, the `get_*` queries and `parameter_file`. Consequently:

- **Overrides of the per-move methods are never called.** `straight_traverse`, `straight_feed`, `straight_probe`, `arc_feed`, `straight_arcsegments`, `rigid_tap`, `dwell`, `user_defined_function`, `set_g5x_offset`, `set_g92_offset`, `set_xy_rotation`, `tool_offset`, `set_plane`, `select_plane`, `set_feed_rate` and `set_spindle_rate` no longer exist on `GLCanon`, and a subclass that defines them is not called back. Read the finished program from `canon.program_geometry` instead.
- **`next_line` is not a per-line tick.** A progress bar overrides `renderer_progress(lineno)`, which fires on the parser's 100 ms tick and before each forwarded callback. AXIS and QtVCP show `(AXIS,notify)` / `(PREVIEW,notify)` messages by checking once more after `load_preview` returns, since no `next_line` follows the comment.
- **Parse-state attributes are gone.** `lo`, `first_move`, `xo`..`wo`, `suppress`, `in_arc`, `plane`, `feedrate`, `g5x_index`, `g5x_offset_*`, `g92_offset_*`, `rotation_xy`, `rotation_sin`, `rotation_cos` and `rotate_and_translate()`. `GLCanon` no longer mixes in `Translated` or `ArcsToSegmentsMixin`; the renderer keeps its own copy of the offsets, the rotation, the plane and the feed rate and forwards none of them. Nothing in the tree reads them - the DROs read the status channel.
- **The per-move lists are gone.** `traverse`, `feed`, `arcfeed`, `moves`, `move_cats` and `preview_zero_rxy` raise `AttributeError` on read, naming the replacement: the program record's `positions()`/`lines`/`kinds`, `g0_length`/`g1_length`/`run_time()` and `extents_zero_rxy`.
- **`tool_list` and `dwells` fill at the end of the parse** (in `adopt_geometry`) rather than growing during it. `dwells` keeps its column order and raw machine coordinates.
- **Still honoured:** `arcdivision` (set from `[DISPLAY]ARCDIVISION`), the `comment` vocabulary (`stop`, `notify`, the foam Z levels; `hide`/`show` are counted in C++ from the same text), and `change_tool`, which the interpreter still needs for a G43 after an M6.

How the preview is put together

The drawing itself lives in `lib/python/rs274/glcanon_scene.py`, in four tiers. Nothing here changes what the preview looks like; it is where to start reading if you need to fix or extend one part of it.

- **Parts.** One class per drawing concern - grid, program geometry, extents, bounding box, offsets, small origin, axes, machine-limits box, tool, live backplot, DRO overlay, the `user_plot()` hook. A part draws that one thing and nothing else. Adding a preview element means adding a part and placing it in the scene's order, not editing an existing part.
- **The scene.** `PreviewScene` holds the parts in draw order and runs them. **Visibility is decided by the scene, not the part**: it evaluates each part's `visible(ctx)` and simply does not call a part whose

gate is false, so `draw()` may assume it is visible and must not open with an `if not shown: return`. Gates that couple parts belong to the scene too - the extents-versus-bounding-box either/or, and the translucent compositing `program_alpha` wraps the program in.

- **Primitives.** Services several parts share - a line array, a wireframe box, Hershey vector text, the tool-cone mesh - reached as `ctx.prim`. Letters are a primitive that axes, extents and offsets all use, not a sibling of "axes".
- **Passes.** The preview is three sets of depth/blend state, not an arbitrary order: world geometry, translucent geometry drawn over it at equal depth, and the screen-space overlay. Each part declares its `pass_`; the scene sets the state when the pass changes. A part that needs something else for its own drawing (the tool's constant-alpha blend) restores it afterwards.

Transforms go through a scoped model-view stack: `with ctx.mv.push():` restores the previous transform on exit, including when an exception unwinds through it, so a part cannot leak a transform onto a later one. Offsets, small origin and axes share one such scope (`RelativeCoordGroup`), because the axes are drawn in the offset frame the offsets progressively build.

Parts read a `FrameContext` - an explicit, enumerated list of machine, view and renderer state, built once per frame by `GlCanonDraw` - rather than the widget itself. That is what lets them be tested without a window: build a context by hand, call `part.draw(ctx)`, and assert on the vertices it emitted.

Click-to-select is not a part - it draws nothing to the screen. `Picker` renders the **same** program geometry into an offscreen framebuffer with line numbers encoded as colour and resolves the nearest hit; `GlCanonDraw.select(x, y)` delegates to it. It shares one `ProgramGeometry` with the drawing part, so the pickable geometry and the drawn geometry cannot drift apart.

Setting `GLCANON_DEBUG=1`, the preview's one verbosity switch, raises the `rs274` logger to DEBUG, checks `glGetError` after each pass, logs which parts the scene drew and which it skipped whenever that split changes, and reports depth/blend state a part left behind.

1.6.6. Neue HAL-Komponenten

Nicht-Echtzeit

mdro mqtt-publisher pi500_vfd pmx485-test qtplasmac-cfg2prefs qtplasmac-materials qtplasmac-plasmac2qt qtplasmac-setup sim-torch svd-ps_vfd

Echtzeit

anglejog div2 enum filter_kalman flipflop homecomp limit_axis mesa_uart millturn scaled_s32_sums tof ton

1.6.7. Neue Treiber

Ein Framework wurde eingeführt zur Steuerung von ModBus-Geräten mit den seriellen Ports auf vielen Mesa-Karten. http://linuxcnc.org/docs/2.9/html/drivers/mesa_modbus.html

Ein neuer GPIO-Treiber für alle GPIO, der von der gpiod-Bibliothek unterstützt wird, ist jetzt enthalten:

http://linuxcnc.org/docs/2.9/html/drivers/hal_gpio.html

1.7. Linux FAQ

Dies sind einige grundlegende Linux-Befehle und -Techniken für Linux-Neulinge. Ausführlichere Informationen finden Sie im Internet oder in den Man Pages.

1.7.1. Automatisches Einloggen

Debian

Debian Stretch verwendet standardmäßig die Xfce-Desktopumgebung mit dem lightDM-Displaymanager lightDM. Um eine automatische Anmeldung bei Debian Stretch zu erhalten:

- Verwenden Sie in einem Terminal den folgenden Befehl:

```
$ /usr/sbin/lightdm --show-config
```

- Notieren Sie sich den absoluten Pfad zur Konfigurationsdatei lightdm.conf.
- Bearbeiten Sie diese Datei mit einem reinen Texteditor (gedit, nano usw.) als root.
- Suchen Sie die folgenden Zeilen und heben Sie deren Auskommentierung auf:

```
#autologin-user=
#autologin-user-timeout=0
```

- Set autologin-user=your_user_name
- Speichern und neu starten.

Ubuntu

Wenn Sie LinuxCNC mit der Ubuntu Live-CD installieren, ist die Voreinstellung, dass Sie sich jedes Mal anmelden müssen, wenn Sie den Computer einschalten. Um die automatische Anmeldung zu aktivieren, gehen Sie zu *System* > *Administration* > *Login Window*. Wenn es sich um eine Neuinstallation handelt, kann es eine oder drei Sekunden dauern, bis das Anmeldefenster erscheint. Sie benötigen Ihr Passwort, das Sie bei der Installation verwendet haben, um Zugang zum Fenster "Einstellungen für das Anmeldefenster" zu erhalten. Aktivieren Sie auf der Registerkarte Sicherheit das Kontrollkästchen Automatische Anmeldung aktivieren und wählen Sie einen Benutzernamen aus der Liste (das wären Sie).

1.7.2. Automatisches Starten

Um LinuxCNC automatisch mit Ihrer Konfiguration nach dem Einschalten des Computers starten zu lassen, gehen Sie zu *System* > *Preferences* > *Sessions* > *Startup Applications*, klicken Sie auf Add.

Navigieren Sie zu Ihrer Konfiguration und wählen Sie die .ini-Datei aus. Wenn sich der Dateiauswahldialog schließt, fügen Sie linuxcnc und ein Leerzeichen vor dem Pfad zu Ihrer .ini-Datei hinzu.

Beispiel:

```
linuxcnc /home/mill/linuxcnc/config/mill/mill.ini
```

Die Dokumentation bezieht sich auf Ihre jeweilige .ini-Datei als INI-Datei.

1.7.3. Terminal

Viele Dinge müssen vom Terminal aus erledigt werden, wie das Überprüfen des Kernel-Meldungspuffers mit *dmesg*. Ubuntu und Linux Mint haben ein Tastaturkürzel Strg + Alt + t. Debian Stretch hat keine Tastaturkürzel definiert. Sie können aber leicht mit dem *Configuration Manager* erstellt werden. Die meisten modernen Dateimanager unterstützen die rechte Taste zum Öffnen eines Terminals. Stellen Sie nur sicher, dass Sie mit der rechten Maustaste auf einen leeren Bereich oder ein Verzeichnis und nicht auf einen Dateinamen klicken. Die meisten Betriebssysteme haben das Terminal als einen Menüpunkt, normalerweise unter Zubehör.

1.7.4. Man Pages

Eine Manpage (kurz für Manual Page) ist eine Form der Software-Dokumentation, die man normalerweise unter UNIX oder UNIX-ähnlichen Betriebssystemen wie Linux findet.

Um eine Manpage anzuzeigen, öffnen Sie ein Terminal, um etwas über den Befehl find im Terminalfenster herauszufinden:

```
man find
```

Verwenden Sie die Tasten Bild auf und Bild ab, um die Manpage anzuzeigen, und die Taste Q, um die Anzeige zu beenden.

NOTE Wenn Sie die man-Seite vom Terminal aus aufrufen, erhalten Sie möglicherweise nicht die erwartete man-Seite. Wenn Sie zum Beispiel man abs eingeben, erhalten Sie die C abs und nicht die LinuxCNC abs. Es ist am besten, die LinuxCNC man-Seiten in den HTML-Dokumenten anzusehen.

1.7.5. Module auflisten

Bei der Fehlersuche müssen Sie manchmal eine Liste der geladenen Module erhalten. Geben Sie in ein Terminalfenster ein:

```
lsmod
```

Wenn Sie die Ausgabe von lsmod in eine Textdatei in einem Terminalfenster senden wollen, geben Sie

ein:

```
lsmod > mymod.txt
```

Die resultierende Textdatei befindet sich im Home-Verzeichnis, wenn Sie beim Öffnen des Terminal-Fensters das Verzeichnis nicht gewechselt haben, und trägt den Namen mymod.txt oder den von Ihnen gewählten Namen.

1.7.6. Bearbeiten einer root-Datei

Wenn Sie den Dateibrowser öffnen und sehen, dass der Eigentümer der Datei root ist, müssen Sie zusätzliche Schritte unternehmen, um diese Datei zu bearbeiten. Die Bearbeitung einiger root-Dateien kann zu schlechten Ergebnissen führen. Seien Sie vorsichtig, wenn Sie root-Dateien bearbeiten. Im Allgemeinen können Sie die meisten root-Dateien öffnen und anzeigen, aber sie bleiben schreibgeschützt.

Der Weg über die Befehlszeile

Öffnen Sie ein Terminal und geben Sie ein

```
sudo gedit
```

Öffnen Sie die Datei mit Datei > Öffnen > Bearbeiten

Der GUI-Weg

1. Klicken Sie mit der rechten Maustaste auf den Desktop und wählen Sie Startprogramm erstellen (engl. Create Launcher).
2. Geben Sie einen Namen ein wie sudo edit.
3. Geben Sie *gksudo "gnome-open %u"* als Befehl ein und speichern Sie den Launcher auf Ihrem Desktop.
4. Ziehen Sie eine Datei auf Ihren Launcher, um sie zu öffnen und zu bearbeiten.

Root-Zugriff

In Ubuntu können Sie root werden, indem Sie "sudo -i" in einem Terminal-Fenster eingeben und dann Ihr Passwort eintippen. Seien Sie vorsichtig, denn als superuser (root) können Sie wirklich alles vermasseln, wenn Sie nicht wissen, was Sie tun.

1.7.7. Terminal-Befehle

Arbeitsverzeichnis

Um den Pfad zum aktuellen Arbeitsverzeichnis herauszufinden, geben Sie im Terminalfenster ein:

```
pwd
```

Wechseln des Arbeitsverzeichnisses (engl. directory)

Um das Arbeitsverzeichnis in das eine Ebene höher liegende Verzeichnis, d.h. das übergeordnete Verzeichnis, zu wechseln, geben Sie im Terminalfenster ein:

```
cd ..
```

Um im Terminalfenster eine Ebene höher zu gehen, geben Sie ein:

```
cd ../..
```

Um direkt in Ihr Heimatverzeichnis zu wechseln, geben Sie im Terminalfenster den Befehl cd ohne Argumente ein:

```
cd
```

Um in das Unterverzeichnis linuxcnc/configs zu gelangen, geben Sie im Terminalfenster ein:

```
cd linuxcnc/configs
```

Auflisten von Dateien in einem Verzeichnis

Um eine Liste aller Dateien und Unterverzeichnisse im Terminalfenster anzuzeigen, geben Sie ein:

```
dir
```

oder

```
ls
```

Suchen einer Datei

Der Befehl find kann für einen neuen Linux-Benutzer etwas verwirrend sein. Die grundlegende Syntax ist:

```
find starting-directory Parameter Aktionen
```

Um zum Beispiel alle .ini-Dateien in Ihrem linuxcnc-Verzeichnis zu finden, müssen Sie zuerst den Befehl pwd verwenden, um das Verzeichnis herauszufinden.

Öffnen Sie ein neues Terminal und geben Sie ein:

```
pwd
```

Und pwd könnte das folgende Ergebnis liefern:

```
/home/joe
```

Mit diesen Informationen setzen Sie den Befehl wie folgt zusammen:

```
find /home/joe/linuxcnc -name \*.ini -print
```

Die Option -name ist der Name der gesuchten Datei, und die Option -print bewirkt, dass das Ergebnis im Terminalfenster ausgegeben wird. Mit *.ini wird find angewiesen, alle Dateien mit der Erweiterung .ini auszugeben. Der Backslash wird benötigt, um den "*" als nicht als Shell-Meta-Zeichen zu interpretieren (engl. Flucht vor Interpretation: "escape"). Weitere Informationen zu find finden Sie in der Manpage find.

Suche nach Text

```
grep -irl 'text to search for' *
```

Dies findet alle Dateien, die den *zu suchenden Text* enthalten, im aktuellen Verzeichnis und allen Unterverzeichnissen darunter, wobei die Groß- und Kleinschreibung ignoriert wird. Die Option -i steht für Ignorieren der Groß- und Kleinschreibung und die Option -r für Rekursiv (schließt alle Unterverzeichnisse in die Suche ein). Die Option -l gibt eine Liste der Dateinamen zurück, wenn Sie die Option -l auslassen, erhalten Sie auch den Text, in dem jedes Vorkommen des zu suchenden Textes gefunden wird. Der * ist ein Platzhalter für die Suche in allen Dateien. Weitere Informationen finden Sie in der Manpage zu grep.

Diagnosemeldungen

Um die Diagnosemeldungen anzuzeigen, verwenden Sie "dmesg" im Befehlsfenster. Um die Diagnosemeldungen in einer Datei zu speichern, verwenden Sie den Umleitungsoperator >, etwa so:

```
dmesg > bootmsg.txt
```

Der Inhalt dieser Datei kann kopiert und online eingefügt werden, um ihn mit Personen zu teilen, die Ihnen bei der Diagnose Ihres Problems helfen.

Um den Nachrichtenpuffer zu löschen, geben Sie Folgendes ein:

```
sudo dmesg -c
```

Dies kann kurz vor dem Start von LinuxCNC hilfreich sein, so dass es nur eine Aufzeichnung von Informationen im Zusammenhang mit dem aktuellen Start von LinuxCNC gibt.

Um die eingebaute Parallelport-Adresse zu finden, verwenden Sie grep, um die Informationen aus dmesg herauszufiltern.

Öffnen Sie nach dem Hochfahren ein Terminal und geben Sie ein:

```
dmesg|grep parport
```

1.7.8. Bequemlichkeiten

Terminal Launcher

Wenn Sie der Bedienfeldleiste am oberen Rand des Bildschirms einen Terminal-Launcher hinzufügen möchten, können Sie normalerweise mit der rechten Maustaste auf das Bedienfeld am oberen Rand des Bildschirms klicken und "Zum Bedienfeld hinzufügen" auswählen. Wählen Sie Custom Application Launcher und Add. Geben Sie der Anwendung einen Namen und geben Sie gnome-terminal in das Befehlsfeld ein.

1.7.9. Hardware-Probleme

Hardware-Informationen

Um herauszufinden, welche Hardware an Ihre Hauptplatine angeschlossen ist, geben Sie in einem Terminalfenster ein:

```
lspci -v
```

Monitor-Auflösung

Während der Installation versucht Ubuntu, die Monitoreinstellungen zu erkennen. Wenn dies fehlschlägt, wird ein allgemeiner Monitor mit einer maximalen Auflösung von 800x600 verwendet.

Eine Anleitung zur Behebung dieses Problems finden Sie hier:

https://help.ubuntu.com/community/FixVideoResolutionHowto

1.7.10. Pfade

Relative Pfade

Relative Pfade basieren auf dem Startverzeichnis, d.h. das Verzeichnis mit der INI-Datei. Die Verwendung relativer Pfade kann das Verschieben von Konfigurationen erleichtern, erfordert aber ein gutes Verständnis der Linux-Pfadangaben.

```
./f0 ist dasselbe wie f0, z. B. eine Datei namens f0 im Startverzeichnis
../f1 bezieht sich auf eine Datei f1 im übergeordneten Verzeichnis
../../f2 bezieht sich auf eine Datei f2 im übergeordneten Verzeichnis des übergeordneten
Verzeichnisses
../../../f3 usw.
```

Chapter 2. Allgemeine Informationen für Anwender

2.1. Vorwort für Anwender

LinuxCNC ist modular und flexibel. Diese Eigenschaften führen dazu, dass viele es als ein verwirrendes Durcheinander von kleinen Dingen sehen und sich fragen, warum es so ist, wie es ist. Diese Seite versucht, diese Frage zu beantworten, bevor Sie in das Dickicht der Dinge zu bekommen.

Die Entwicklung von LinuxCNC begann am National Institute of Standards and Technology in den USA. Es wuchs mit UNIX als Betriebssystem auf. UNIX machte es anders. Unter den frühen UNIX-Entwicklern etablierten sich eine Reihe von Ideen für das Schreiben von Code, die einige als den UNIX-Weg bezeichnen. Diese LinuxCNC-Autoren folgten diesen Wegen.

Eric S. Raymond fasst in seinem Buch The Art of UNIX Programming die UNIX-Philosophie als die weit verbreitete technische Philosophie "Keep it Simple, Stupid" (KISS-Prinzip) zusammen. Er beschreibt dann, wie diese allgemeine Philosophie seiner Meinung nach als kulturelle UNIX-Norm angewandt wird, obwohl es nicht überraschend ist, dass man in der tatsächlichen UNIX-Praxis schwerwiegende Verstöße gegen die meisten der folgenden Punkte finden kann:

- Regel der Modularität: Schreibe einfache Teile, die durch saubere Schnittstellen verbunden sind.
- Regel der Klarheit: Klarheit ist besser als Cleverness.
- Regel der Komposition: Entwerfen Sie Programme so, dass sie mit anderen Programmen verbunden sind.
- Trennungsgrundsatz: Trenne die formalisierte Prinzipien (engl. policy) vom Mechanismus; trenne die Schnittstellen von den Motoren. [1]

Herr Raymond bot einige weitere Regeln an, aber diese vier beschreiben wesentliche Merkmale des LinuxCNC Motion Control Systems.

Die **Modularitätsregel** ist entscheidend. In diesen Handbüchern wird immer wieder vom Interpreter, Taskplaner, Motion oder HAL gesprochen. Jedes dieser Elemente ist ein Modul oder eine Sammlung von Modulen. Es ist die Modularität, die es Ihnen ermöglicht, genau die Teile zusammenzufügen, die Sie zum Betrieb Ihrer Maschine benötigen.

Die **Clarity** Regel ist wesentlich. An LinuxCNC wird weiter gearbeitet - es ist nicht fertig, und wird es niemals sein. Es ist vollständig genug, um die meisten Maschinen zu laufen, auf den wir wollen, dass es läuft. Ein großer Teil dieses Fortschritts wurde erreicht, weil viele Benutzer und Code-Entwickler in der Lage sind, sich die Arbeit anderer anzusehen und auf dem aufzubauen, was sie getan haben.

Die **Zusammensetzungs**-Regel ermöglicht es uns, aus den vielen verfügbaren Modulen ein vorhersehbares Steuerungssystem zu erstellen, indem wir sie miteinander verbinden können. Wir erreichen die Anschlussfähigkeit, indem wir Standardschnittstellen zu Modulgruppen einrichten und diesen Standards folgen.

Die **Trennungsregel** verlangt, dass wir verschiedene Teile erstellen, die kleine Dinge tun. Durch die Trennung von Funktionen ist die Fehlersuche viel einfacher, und Ersatzmodule können in das System eingefügt und leicht verglichen werden.

Was bedeutet der UNIX-Weg für Sie als Benutzer von LinuxCNC. Es bedeutet, dass Sie in der Lage sind, Entscheidungen darüber zu treffen, wie Sie das System verwenden werden. Viele dieser Entscheidungen sind ein Teil der Maschinenintegration, aber viele beeinflussen auch die Art und Weise, wie Sie Ihre Maschine benutzen werden. Beim Lesen werden Sie viele Stellen finden, an denen Sie Vergleiche anstellen müssen. Schließlich werden Sie sich entscheiden: "Ich verwende lieber diese Schnittstelle als jene" oder "Ich schreibe Teileverschiebungen lieber auf diese Weise als auf jene". In diesem Handbuch beschreiben wir die ganze Breite der sich heute bereits bietenden Möglichkeiten.

Wie Sie Ihre Reise in die Verwendung von LinuxCNC beginnen, bieten wir zwei warnende Hinweise:[2]

- Um es mit den Worten von Doug Gwyn über UNIX zu sagen: "LinuxCNC wurde nicht entwickelt, um seine Benutzer davon abzuhalten, dumme Dinge zu tun, denn das würde sie auch davon abhalten, kluge Dinge zu tun."
- Auch die Worte von Steven King: "LinuxCNC ist benutzerfreundlich. Es ist nur nicht freizügig, mit welchen Benutzern es freundlich ist."

Eine Reihe von Videos auf YouTube zeigen, dass ein Umstieg auf LinuxCNC möglich ist, unabhängig davon, welches Betriebssystem Sie benutzen. Das heißt, mit dem Aufkommen der additiven Fertigung wie 3D-Druck gibt es ein zunehmendes Interesse von der breiteren IT-Gemeinschaft in CNC-Bearbeitung und es sollte möglich sein, jemanden mit komplementären Fähigkeiten / Ausrüstung in Ihrer Nähe zu finden, um gemeinsam die anfänglichen Hürden zu überwinden.

2.2. LinuxCNC Anwender-Einführung

2.2.1. Einführung

Dieses Dokument konzentriert sich auf die Verwendung von LinuxCNC, es ist für Leser gedacht, die es bereits installiert und konfiguriert haben. Einige Informationen zur Installation werden in den folgenden Kapiteln gegeben. Die vollständige Dokumentation über die Installation und Konfiguration kann im Handbuch des Integrators gefunden werden.

2.2.2. Wie LinuxCNC funktioniert

LinuxCNC ist eine Reihe von hochgradig anpassbaren Anwendungen für die Steuerung von computergesteuerten Fräsmaschinen und Drehmaschinen, 3D-Druckern, Robotern, Laserschneidern, Plasmaschneidern und anderen automatisierten Geräten. Sie ist in der Lage, eine koordinierte Steuerung von bis zu 9 Bewegungsachsen zu ermöglichen.

Im Kern besteht LinuxCNC aus mehreren Schlüsselkomponenten, die zusammen integriert sind, um ein komplettes System zu bilden:

- eine grafische Benutzeroberfläche (GUI) als grundlegende Schnittstelle zwischen dem Bediener, der Software und der CNC-Maschine selbst bildet;
- der Hardware Abstraction Layer (HAL) bietet eine Methode zur Verknüpfung aller verschiedenen internen virtuellen Signale, die von LinuxCNC erzeugt und empfangen werden, mit der Außenwelt,

und

- die High-Level-Controller, welche die Erzeugung und Ausführung der Bewegungssteuerung der CNC-Maschine koordinieren, sind der Motion Controller (EMCMOT), der diskrete Input/Output-Controller (EMCIO) und der Task Executor (EMCTASK).

Die nachfolgende Darstellung ist ein einfaches Blockdiagramm, das zeigt, wie eine typische 3-Achsen-CNC-Fräse mit Schrittmotoren aussehen könnte:

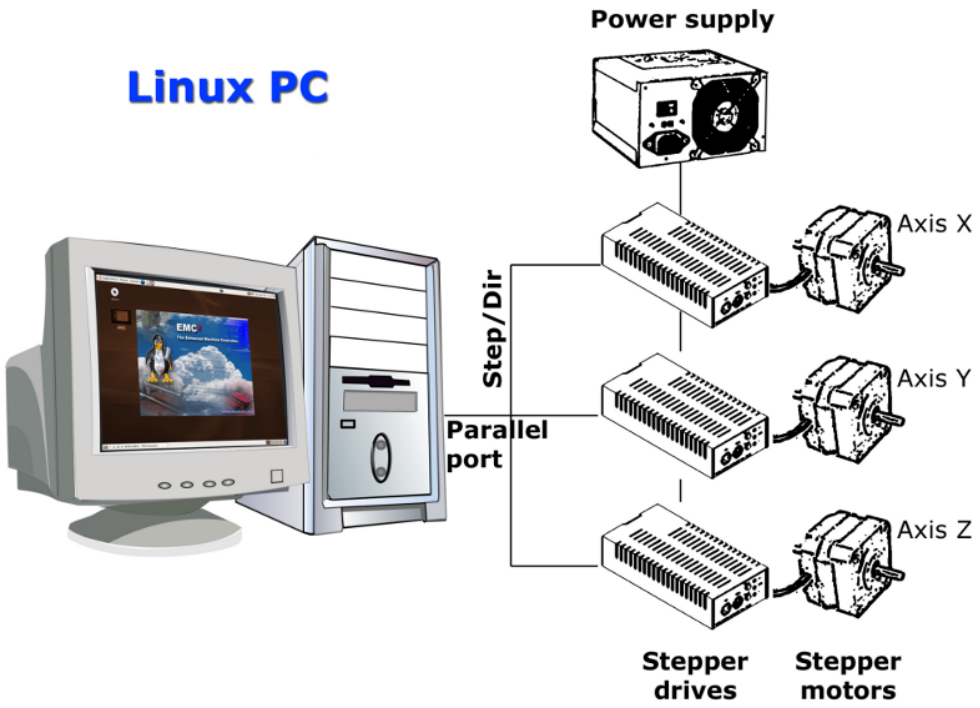


Figure 4. Einfache LinuxCNC-gesteuerte Maschine

Ein Computer, auf dem LinuxCNC läuft, sendet eine Folge von Impulsen über die parallele Schnittstelle an die Schrittantriebe, von denen jeder einen Schrittmotor hat, der an ihn angeschlossen ist. Jeder Antrieb erhält zwei unabhängige Signale; ein Signal, um den Antrieb zu befehlen, seinen zugehörigen Schrittmotor im oder gegen den Uhrzeigersinn zu bewegen, und ein zweites Signal, das die Geschwindigkeit definiert, mit der sich dieser Schrittmotor dreht.

Während ein Schrittmotor-System unter parallelen Port-Steuerung dargestellt ist, kann ein LinuxCNC-System auch die Vorteile einer Vielzahl von dedizierten Hardware-Motion-Control-Schnittstellen für erhöhte Geschwindigkeit und E / A-Funktionen. Eine vollständige Liste der von LinuxCNC unterstützten Schnittstellen kann auf der unterstützte Hardware Seite des Wikis gefunden werden.

In den meisten Fällen werden die Benutzer eine Konfiguration speziell für ihre Mühle Setup mit entweder der Stepper Configuration Wizard (für CNC-Systeme, die mit dem Computer 'Parallel-Port) oder die Mesa Hardware Wizard (für fortgeschrittene Systeme mit einem Mesa I/O PCI-Karte). Wenn Sie einen der beiden Assistenten ausführen, werden mehrere Ordner auf der Festplatte des Computers erstellt, die eine Reihe von Konfigurationsdateien enthalten, die für diese CNC-Maschine spezifisch sind, und ein Symbol wird auf dem Desktop platziert, um den einfachen Start von LinuxCNC zu ermöglichen.

Wenn zum Beispiel der Assistent für die Konfiguration von Schrittmotoren verwendet wurde, um eine Einrichtung für die oben abgebildete 3-Achsen-CNC-Fräse mit dem Namen "My_CNC" zu erstellen,

würden die vom Assistenten erstellten Ordner in der Regel die folgenden Dateien enthalten:

- **Ordner:** `My_CNC`
  - `Meine_CNC.ini`
    Die INI-Datei enthält alle grundlegenden Hardware-Informationen für den Betrieb der CNC-Fräse, wie z.B. die Anzahl der Schritte, die jeder Schrittmotor drehen muss, um eine volle Umdrehung zu vollenden, die maximale Geschwindigkeit, mit der jeder Schrittmotor Geschwindigkeit, mit der jeder Schrittmotor arbeiten darf, die Verfahrgrenzen jeder Achse oder die Konfiguration und das Verhalten der Endschalter an jeder Achse.
  - `My_CNC.hal`
    Diese HAL-Datei enthält Informationen, die LinuxCNC mitteilen, wie die internen virtuellen Signale mit physischen Verbindungen außerhalb des Computers zu verbinden. Zum Beispiel, die Angabe von Pin 4 an der parallelen Schnittstelle zu senden die Z-Achse Schritt Richtung Signal, oder Anweisung LinuxCNC zu stoppen Fahren der X-Achse Motor, wenn ein Endschalter Schalter am Parallelport Pin 13 ausgelöst wird.
  - `custom.hal`
    Anpassungen der Fräsenenkonfiguration, die über den Umfang des Assistenten hinausgehen, können vorgenommen werden, indem man weitere Links zu anderen virtuellen Punkten innerhalb von LinuxCNC in dieser HAL-Datei. Beim Starten einer LinuxCNC-Sitzung wird diese Datei gelesen und verarbeitet, bevor die GUI geladen wird. Ein Beispiel kann die Initiierung der Modbus Kommunikation mit dem Spindelmotor, so dass er als betriebsbereit bestätigt wird, bevor die GUI angezeigt wird.
  - `custom_postgui.hal`
    Die custom_postgui HAL-Datei erlaubt eine weitere Anpassung von LinuxCNC, unterscheidet sich aber von custom.HAL darin, dass sie verarbeitet wird, nachdem die GUI angezeigt wurde. Zum Beispiel, nach dem Aufbau der Modbus-Kommunikation zu den Spindelmotor in custom.hal, kann LinuxCNC die custom_postgui Datei verwenden, um die Spindeldrehzahl Auslese vom Motorantrieb mit einem auf der GUI angezeigten Bargraph zu verbinden.
  - `postgui_backup.hal`
    Diese Datei wird als Sicherungskopie der Datei custom_postgui.hal bereitgestellt, um dem Benutzer die Möglichkeit zu geben, schnell eine postgui-HAL-Konfiguration wiederherzustellen. Dies ist besonders nützlich, wenn der Benutzer den Konfigurationsassistenten erneut unter demselben Namen *My_CNC* ausführen möchte, um einige Parameter der Fräse zu ändern. Das Speichern der Mühlenkonfiguration im Assistenten überschreibt die bestehende Datei custom_postgui und lässt die postgui_backup Datei unberührt.
  - `tool.tbl`
    Eine Werkzeugtabellendatei enthält eine parametrisierte Liste aller von der Fräse verwendeten Schneidwerkzeuge. Diese Parameter können Fräserdurchmesser und -länge enthalten und werden verwendet, um einen Katalog von Daten bereitzustellen, der LinuxCNC sagt, wie seine Bewegung für unterschiedlich große Werkzeuge innerhalb einer Fräsoperation kompensieren soll.
- **Ordner:** `nc_files`
  Der Ordner "nc_files" dient als Standardspeicherort für die G-Code-Programme zur Steuerung der Fräse. Er enthält auch eine Reihe von Unterordnern mit G-Code-Beispielen.

2.2.3. Grafische Benutzeroberfläche (engl. Graphical User Interface)

Eine grafische Benutzeroberfläche ist der Teil der LinuxCNC, mit dem der Werkzeugmaschinenbediener interagiert. LinuxCNC verfügt über verschiedene Arten von Benutzeroberflächen, aus denen Sie auswählen können, indem bestimmte Felder bearbeitet werden, die in der Datei INI enthalten sind:

ACHSE

AXIS, die Standard-Tastatur-GUI-Schnittstelle. Dies ist auch die Standard-GUI, die gestartet wird, wenn ein Konfigurationsassistent zum Erstellen eines Desktop-Symbol-Launchers verwendet wird:

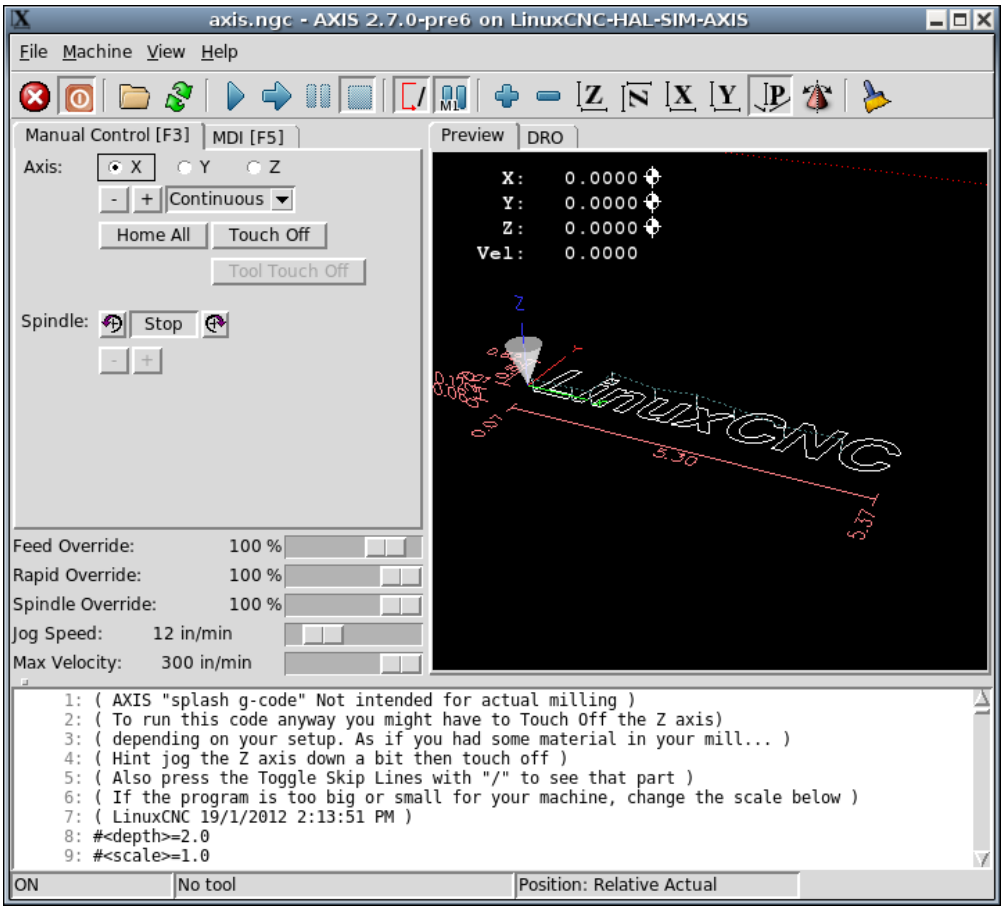


Figure 5. AXIS, die standardmäßige Tastatur-GUI-Schnittstelle

Touchy

Touchy, eine GUI für Touchscreens:

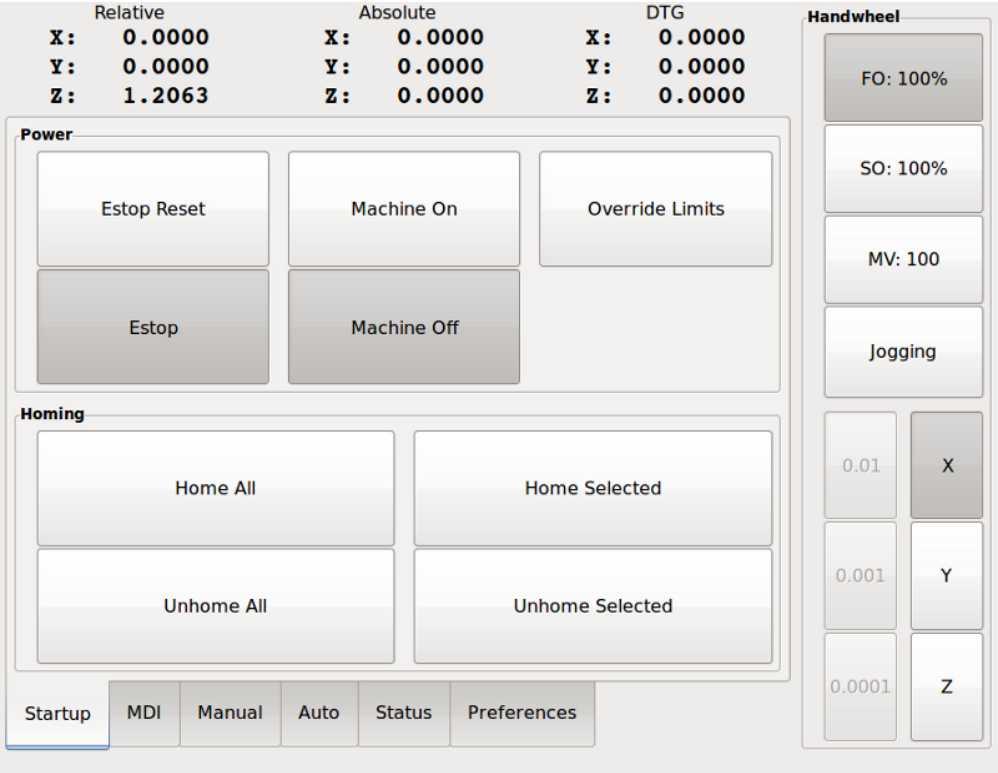


Figure 6. Touchy, eine Touchscreen-GUI

Gscreen

Gscreen, eine vom Benutzer konfigurierbare Touchscreen-GUI:

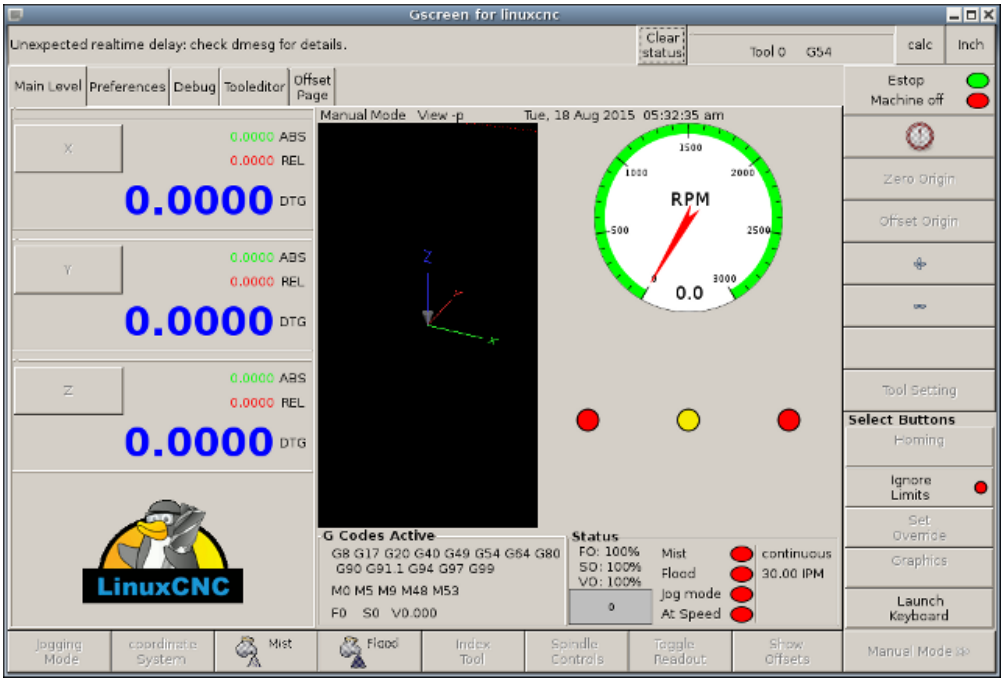


Figure 7. Gscreen, eine konfigurierbare Basis-Touchscreen-GUI

GMOCCAPY

GMOCCAPY, eine Touchscreen-GUI, die auf Gscreen basiert. GMOCCAPY ist so konzipiert, dass es auch in Anwendungen funktioniert, bei denen Tastatur und Maus die bevorzugten Methoden zur Steuerung der GUI sind:

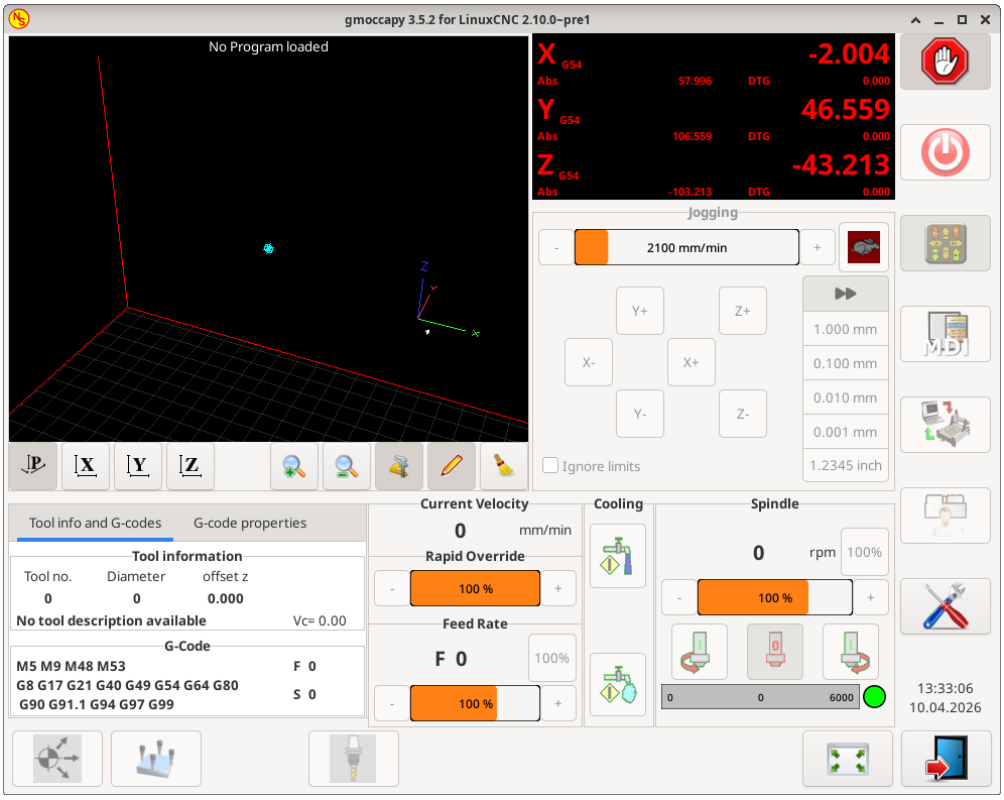


Figure 8. GMOCCAPY, eine Touchscreen-GUI auf der Grundlage von Gscreen

NGCGUI

NGCGUI, ein Subroutinen-GUI, das die Programmierung von G-Code im Assistentenstil ermöglicht. NGCGUI kann als eigenständiges Programm ausgeführt oder in eine andere GUI als eine Reihe von Registerkarten eingebettet werden. Der folgende Screenshot zeigt NGCGUI eingebettet in AXIS:

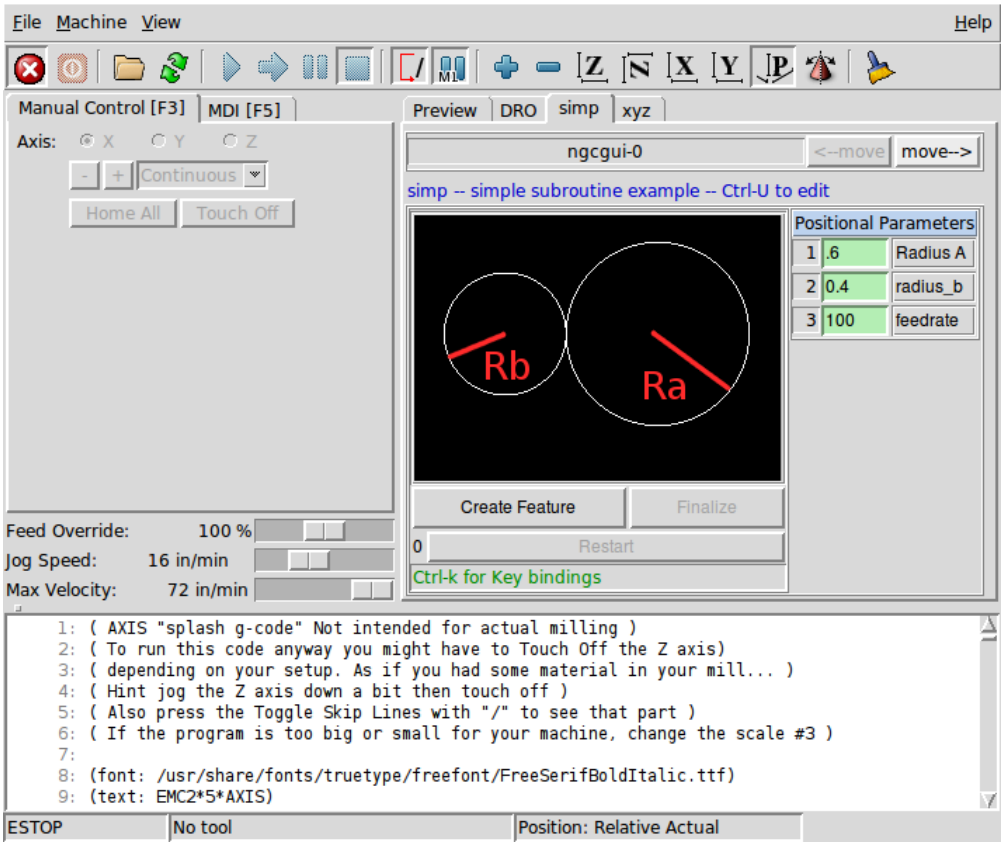


Figure 9. NGCGUI, eine in AXIS integrierte grafische Benutzeroberfläche

TkLinuxCNC

TkLinuxCNC, eine weitere Schnittstelle, die auf Tcl/Tk basiert. Ehemals die beliebteste Schnittstelle nach AXIS.

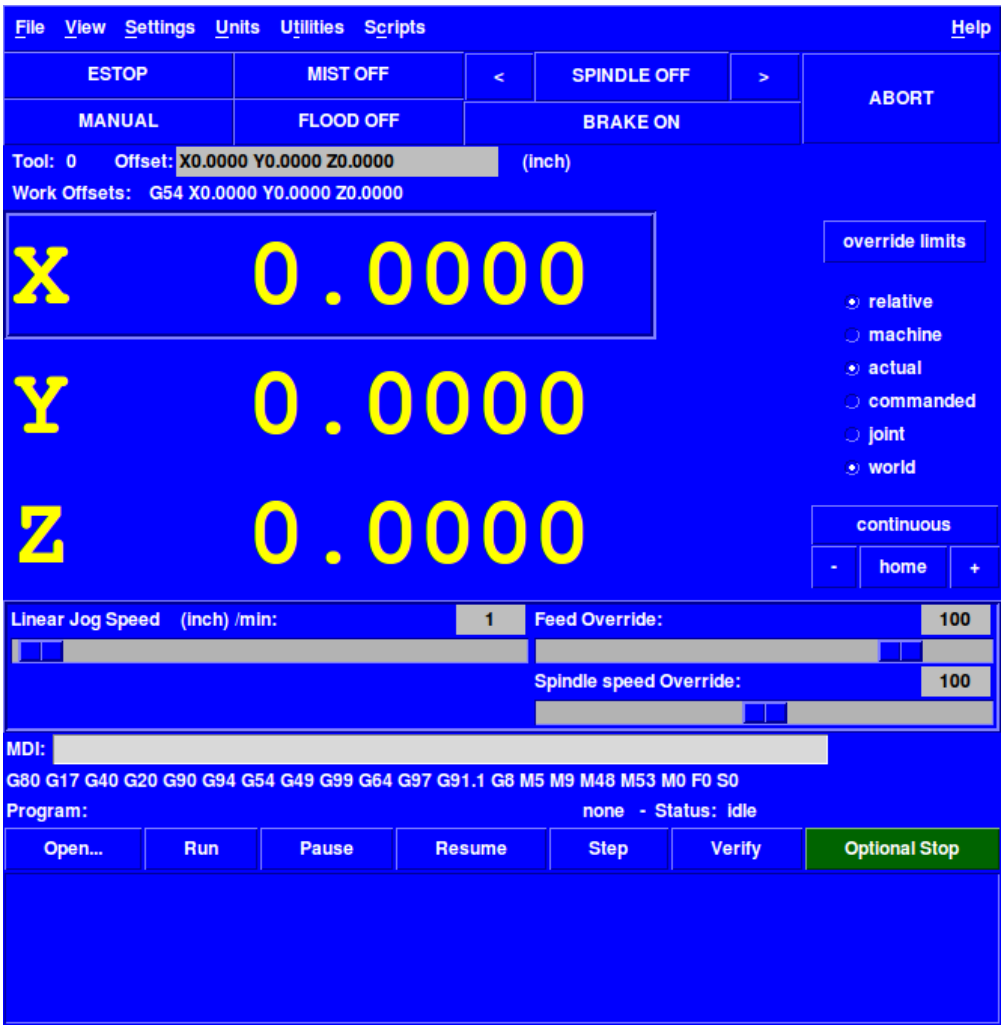


Figure 10. TkLinuxCNC grafische Schnittstelle

QtDragon

QtDragon, ein Touchscreen GUI basierend auf QtVCP mit der PyQt5 Bibliothek. Sie kommt in den Versionen *QtDragon* und *QtDragon_hd*. Sie sind sehr ähnlich in Features, aber QtDragon_hd ist für größere Monitore vorbereitet.



Figure 11. QtDragon, eine Touchscreen-GUI auf der Grundlage von QtVCP

QtPlasmaC

QtPlasmaC, ein Touchscreen-Plasmaschnitt GUI basierend auf QtVCP mit der PyQt5-Bibliothek. Es kommt in drei Aspektverhältnissen, 16:9, 4:3, und 9:16.

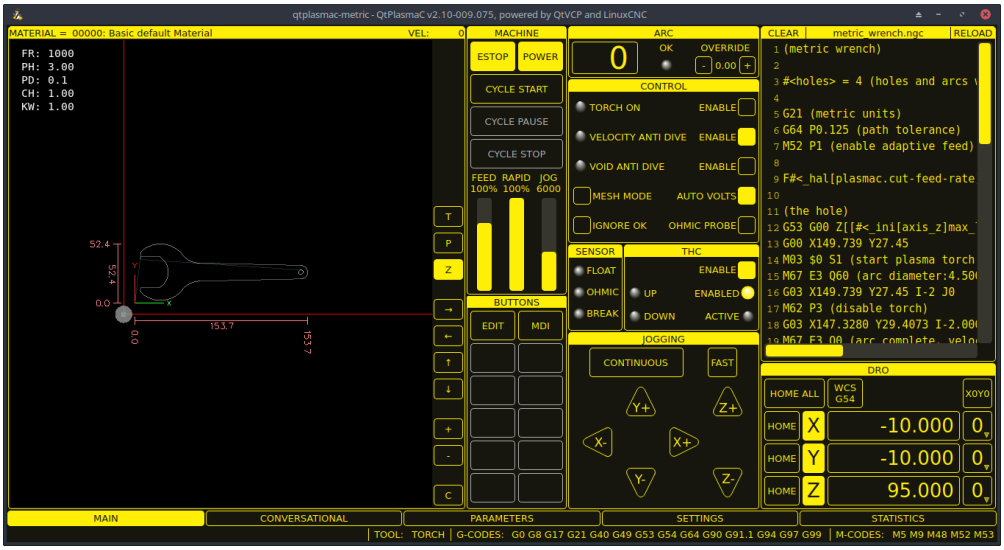


Figure 12. QtPlasmaC, eine Touchscreen-GUI zum Plasma-Schneiden auf der Grundlage von QtVCP

2.2.4. Benutzerschnittstellen

Diese Benutzeroberflächen sind eine Möglichkeit, mit LinuxCNC außerhalb der grafischen Benutzeroberflächen zu interagieren.

halui

Eine HAL-basierte Benutzeroberfläche, die es erlaubt, LinuxCNC mit Tasten und Schaltern zu steuern

linuxcncrsh

Eine Telnet-basierte Benutzeroberfläche, die es ermöglicht, Befehle von entfernten Computern zu senden.

2.2.5. Virtuelle Schalttafeln

Wie oben erwähnt, können viele der GUIs von LinuxCNC vom Benutzer angepasst werden. Dies kann geschehen, um Indikatoren, Anzeigen, Schalter oder Schieberegler zum grundlegenden Aussehen einer der GUIs für erhöhte Flexibilität oder Funktionalität hinzuzufügen. Zwei Arten von Virtual Control Panel sind in LinuxCNC angeboten:

PyVCP

PyVCP, ein auf Python basierendes virtuelles Bedienfeld, das der AXIS-GUI hinzugefügt werden kann. PyVCP nutzt nur virtuelle Signale, die im Hardware Abstraction Layer enthalten sind, wie z.B. die Spindel-bei-Drehzahl-Anzeige oder das Notstopp-Ausgangssignal, und hat ein einfaches, schnörkelloses Aussehen. Dies macht es zu einer ausgezeichneten Wahl, wenn der Benutzer ein virtuelles Bedienfeld mit minimalem Aufwand hinzufügen möchte.

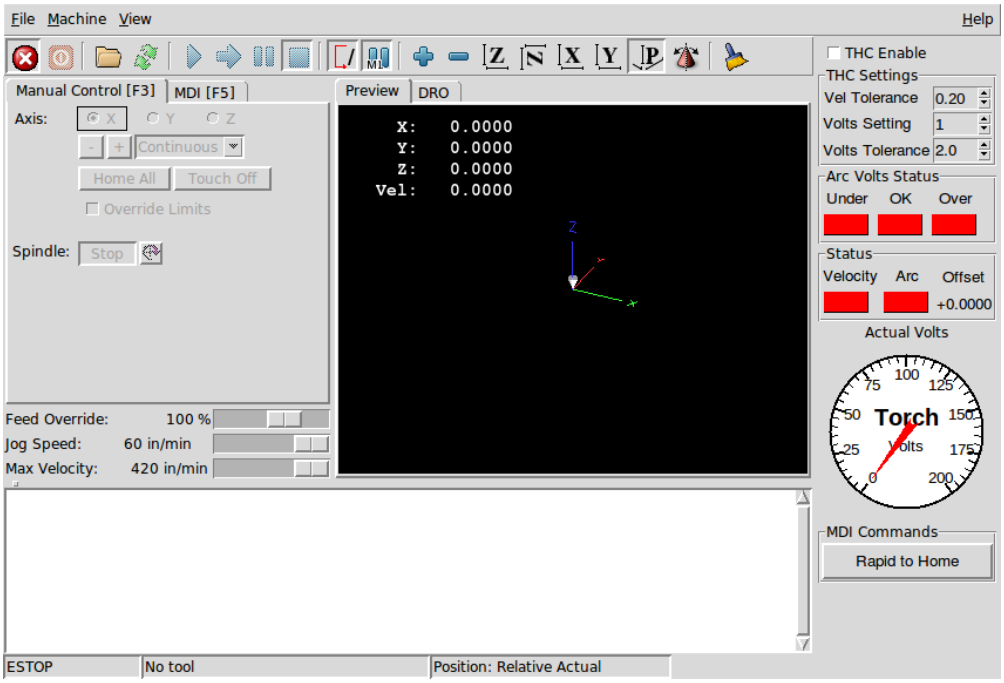


Figure 13. PyVCP-Beispiel eingebettet in AXIS GUI

GladeVCP

GladeVCP, ein Glade-basiertes virtuelles Bedienfeld, das zu den AXIS oder Touchy GUIs hinzugefügt werden kann. GladeVCP hat den Vorteil gegenüber PyVCP, dass es nicht auf die Anzeige oder Steuerung von virtuellen HAL-Signalen beschränkt ist, sondern auch andere externe Schnittstellen außerhalb von LinuxCNC wie Fenster oder Netzwerkereignisse einbeziehen kann. GladeVCP ist auch flexibler in der Art und Weise, wie es konfiguriert werden kann, um auf der GUI zu erscheinen:

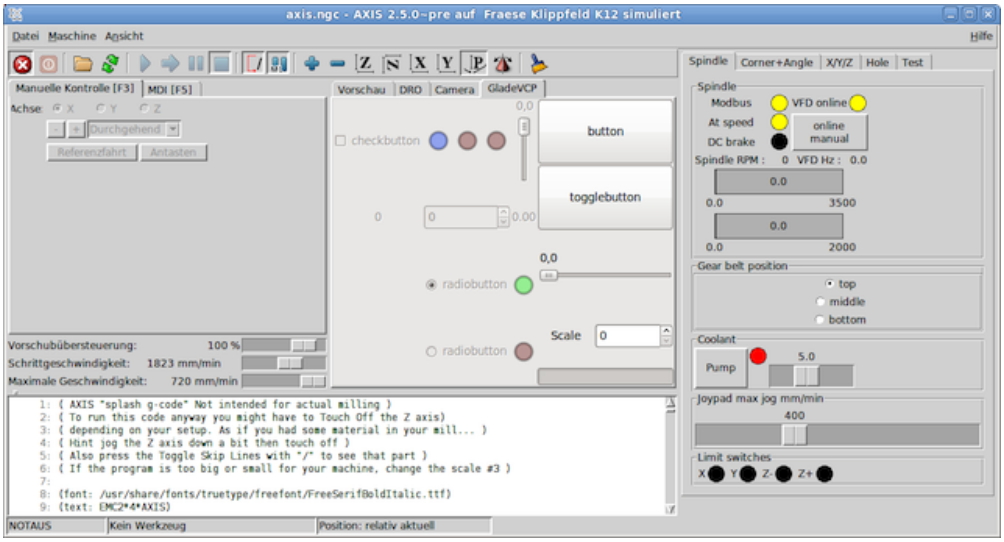


Figure 14. GladeVCP-Beispiel eingebettet in AXIS GUI

QtVCP

QtVCP, ein auf PyQt5 basierendes virtuelles Bedienfeld, das zu den meisten GUIs hinzugefügt oder als eigenständiges Bedienfeld ausgeführt werden kann. QtVCP hat gegenüber PyVCP den Vorteil, dass es nicht auf die Anzeige oder Steuerung von virtuellen HAL-Signalen beschränkt ist, sondern auch andere externe Schnittstellen außerhalb von LinuxCNC, wie z.B. Fenster- oder Netzwerkereignisse, durch Erweiterung mit Python-Code einbeziehen kann. QtVCP ist auch flexibler, wie es konfiguriert werden kann, um auf der GUI mit vielen speziellen Widgets erscheinen:

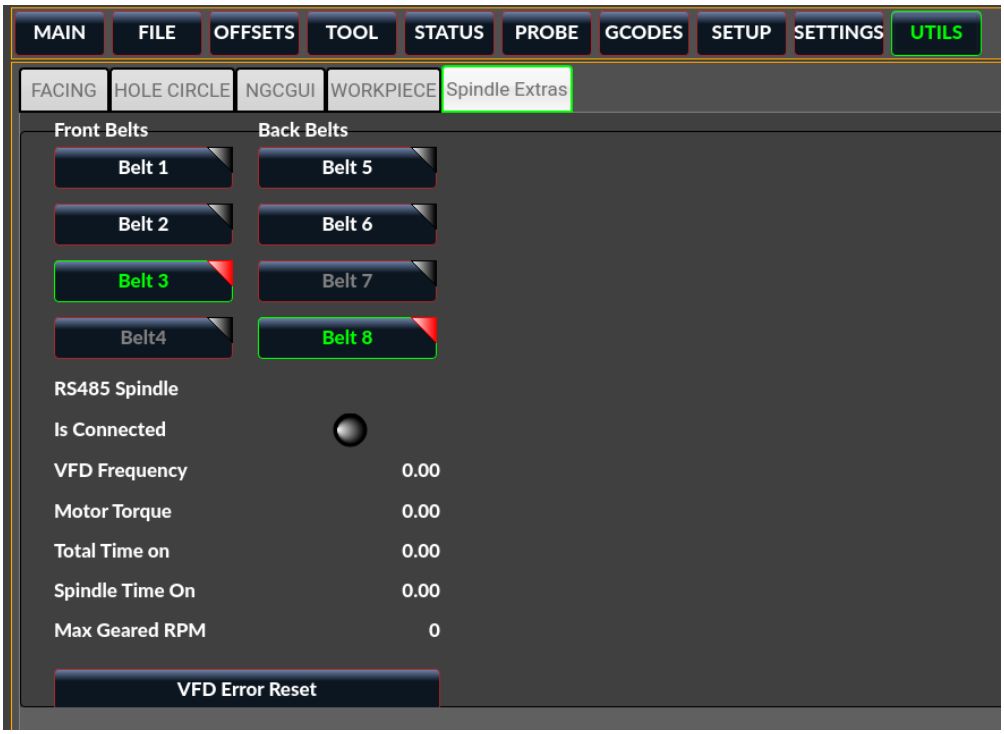


Figure 15. QtVCP-Beispiel eingebettet in QtDragon GUI

2.2.6. Sprachen

LinuxCNC verwendet Übersetzungsdateien, um LinuxCNC Benutzeroberflächen in viele Sprachen zu übersetzen, einschließlich Französisch, Deutsch, Italienisch, Finnisch, Russisch, Rumänisch,

Portugiesisch und Chinesisch. Unter der Annahme, dass eine Übersetzung erstellt wurde, wird LinuxCNC automatisch verwenden, was auch immer Muttersprache Sie sich mit, wenn Sie das Linux-Betriebssystem. Wenn Ihre Sprache nicht übersetzt wurde, wenden Sie sich an einen Entwickler im IRC, in der Mailingliste oder im Benutzerforum, um Hilfe zu erhalten.

2.2.7. Denken wie ein CNC-Bediener

Dieses Handbuch gibt nicht vor, Ihnen den Umgang mit einer Dreh- oder Fräsmaschine beizubringen. Um ein erfahrener Bediener zu werden, braucht man viel Zeit und eine Menge Arbeit. Ein Autor hat einmal gesagt: *Wir lernen durch Erfahrung, wenn man sie macht*. Zerbrochene Werkzeuge, angegriffene Schraubstöcke und die Narben sind der Beweis für die gelernten Lektionen. Eine schöne Oberfläche, enge Toleranzen und Vorsicht bei der Arbeit sind Beweise für gelernte Lektionen. Keine Maschine und kein Programm kann die menschliche Erfahrung ersetzen.

Jetzt, wo Sie anfangen, mit der LinuxCNC-Software zu arbeiten, müssen Sie sich in die Rolle eines Bedieners versetzen. Sie müssen sich in die Rolle einer Person versetzen, die für eine Maschine verantwortlich ist. Es ist eine Maschine, die auf Ihre Befehle wartet und dann die Befehle ausführt, die Sie ihr geben werden. Auf diesen Seiten werden wir die Erklärungen geben, die Ihnen helfen werden, ein guter CNC-Bediener mit LinuxCNC zu werden.

2.2.8. Betriebsarten

Wenn LinuxCNC läuft, gibt es drei verschiedene Haupt-Modi für die Eingabe von Befehlen verwendet. Diese sind Manuell, Auto, und Manuelle Dateneingabe (MDI). Der Wechsel von einem Modus zum anderen macht einen großen Unterschied in der Art und Weise, dass die LinuxCNC-Steuerung verhält. Es gibt bestimmte Dinge, die in einem Modus getan werden können, die in einem anderen nicht getan werden können. Ein Bediener kann eine Achse im manuellen Modus referenzieren, aber nicht im Auto- oder MDI-Modus. Ein Bediener kann die Maschine veranlassen, eine ganze Datei voll von G-Codes in der Auto-Modus, aber nicht in der manuellen oder MDI ausführen.

Im manuellen Modus wird jeder Befehl einzeln eingegeben. In menschlicher Sprache könnte ein manueller Befehl lauten: "Kühlmittel einschalten" oder "X mit 25 Zoll pro Minute verfahren". Dies entspricht in etwa dem Umlegen eines Schalters oder dem Drehen des Handrads für eine Achse. Diese Befehle werden normalerweise auf einer der grafischen Oberflächen durch Drücken einer Schaltfläche mit der Maus oder durch Drücken einer Taste auf der Tastatur ausgeführt. Im Automodus kann eine ähnliche Taste oder ein Tastendruck verwendet werden, um ein ganzes Programm mit G-Code zu laden oder dessen Ausführung zu starten, das in einer Datei gespeichert ist. Im MDI-Modus kann der Bediener einen Codeblock eingeben und die Maschine durch Drücken der <Return>- oder <Enter>-Taste auf der Tastatur anweisen, ihn auszuführen.

Einige Bewegungssteuerungsbefehle sind gleichzeitig verfügbar und führen in allen Modi zu denselben Bewegungsänderungen. Dazu gehören Abbruch, Notaus und Vorschub-Override. Befehle wie diese sollten selbsterklärend sein.

Die AXIS-Benutzeroberfläche verbirgt einige der Unterschiede zwischen Auto und den anderen Modi, indem sie Auto-Befehle in den meisten Fällen verfügbar macht. Sie verwischt auch die Unterscheidung zwischen Manuell und MDI, da einige manuelle Befehle, wie z. B. "Touch Off", eigentlich durch das Senden von MDI-Befehlen implementiert werden. Dies geschieht, indem automatisch in den Modus

gewechselt wird, der für die vom Benutzer angeforderte Aktion erforderlich ist.

2.3. Wichtige Anwendungskonzepte

Dieses Kapitel behandelt wichtige Benutzerkonzepte, die Sie verstehen sollten, bevor Sie versuchen, eine CNC-Maschine mit G-Code zu betreiben.

2.3.1. Trajektorien Steuerung

Trajektorienplanung

Mit "Trajektorien Planung" werden allgemein diejenigen Methoden bezeichnet, die LinuxCNC dem durch Ihren G-Code festgelegten Pfad folgen lassen, eingeschränkt nur durch die Möglichkeiten (limits) Ihrer Maschine.

Ein G-Code-Programm kann nie vollständig befolgt werden. Stellen Sie sich zum Beispiel vor, Sie geben als einzeiliges Programm den folgenden Zug an:

```
G1 X1 F10 (G1 ist die lineare Bewegung, X1 ist das Ziel, F10 ist die Geschwindigkeit)
```

In der Realität kann nicht die gesamte Bewegung mit F10 durchgeführt werden, da die Maschine aus dem Stillstand heraus beschleunigen, sich in Richtung X=1 bewegen und dann abbremsen muss, um wieder anzuhalten. Manchmal wird ein Teil der Bewegung mit F10 ausgeführt, aber bei vielen Bewegungen, insbesondere bei kurzen, wird die angegebene Vorschubgeschwindigkeit überhaupt nicht erreicht. Kurze Bewegungen in Ihrem G-Code können dazu führen, dass Ihre Maschine langsamer und bei längeren Bewegungen schneller wird, wenn der *naive Nockendetektor* nicht mit G64 P*n* verwendet wird.

Die oben beschriebene grundlegende Beschleunigung und Verzögerung ist nicht komplex und es müssen keine Kompromisse eingegangen werden. Die in der INI-Datei angegebenen Maschinenbeschränkungen wie maximale Achsengeschwindigkeit und Achsenbeschleunigung müssen vom Trajektorienplaner eingehalten werden.

Für weitere Informationen zu den Trajektorie-Panner INI-Optionen siehe den Abschnitt zu Trajektorien im INI Kapitel.

Pfad Folgen (engl. path following)

Ein weniger einfaches Problem ist das der Bahnverfolgung. Wenn Sie eine Ecke in G-Code programmieren, kann der Bahnplaner mehrere Dinge tun, die alle in einigen Fällen richtig sind:

- Es kann genau an den Koordinaten der Kurve bis zum Stillstand abbremsen und dann in die neue Richtung beschleunigen.
- Sie kann auch das so genannte Blending durchführen, d. h. die Vorschubgeschwindigkeit beim Durchfahren der Ecke aufrechterhalten, so dass die Ecke abgerundet werden muss, um die

Maschinenvorgaben einzuhalten.

Sie sehen, dass es hier einen Kompromiss gibt: Sie können die Geschwindigkeit verringern, um eine bessere Bahnverfolgung zu erhalten, oder die Geschwindigkeit beibehalten und eine schlechtere Bahnverfolgung haben. Je nach Art des Schnitts, des Materials, des Werkzeugs usw. kann der Programmierer unterschiedliche Kompromisse eingehen.

Auch bei schnellen Bewegungen wird die aktuelle Bahnsteuerung beachtet. Mit Bewegungen, die lang genug sind, um die maximale Geschwindigkeit auf einer Maschine mit geringer Beschleunigung und ohne vorgegebene Bahntoleranz zu erreichen, kann man eine ziemlich runde Ecke erhalten.

Programmierung des Planers

Die Befehle zur Steuerung der Trajektorie lauten wie folgt:

G61

(Exact Path Mode) Der G61 fährt den programmierten Punkt exakt an, auch wenn das bedeutet, dass er vorübergehend zum Stillstand kommt, um die Richtung zum nächsten programmierten Punkt zu ändern.

G61.1

(Exakter Stoppmodus) G61.1 weist den Planer an, einen exakten Stopp an jedem Segmentende zu setzen. Die Bahn wird genau eingehalten, aber vollständige Vorschubstopps können für das Werkstück oder das Werkzeug destruktiv sein, je nach den Besonderheiten der Bearbeitung.

G64

(Blend Without Tolerance Mode) „G64“ ist die Standardeinstellung, wenn Sie LinuxCNC starten. G64 mischt nur und der naive Nockendetektor ist nicht aktiviert. G64 und G64 P0 weisen den Planer an, die Bahnfolgegenauigkeit zu opfern, um die Vorschubgeschwindigkeit aufrechtzuerhalten. Dies ist für einige Arten von Materialien oder Werkzeugen erforderlich, bei denen exakte Stopps schädlich sind, und kann gut funktionieren, solange der Programmierer darauf achtet, dass der Pfad des Werkzeugs etwas kurviger ist als vom Programm angegeben. Wenn Sie G0-Bewegungen (Schnellfahrten) mit G64 verwenden, seien Sie vorsichtig bei Freiraumbewegungen und lassen Sie genügend Abstand, um Hindernisse zu überwinden, basierend auf den Beschleunigungsfähigkeiten Ihrer Maschine.

G64 P- Q-

(Blend With Tolerance Mode) Dies aktiviert den „naiven CAM-Detektor“ und ermöglicht das Abrunden von Ecken (engl. blending) mit einer Toleranz. Wenn Sie G64 P0.05 programmieren, teilen Sie dem Planer mit, dass Sie einen kontinuierlichen Vorschub wünschen, an den programmierten Ecken jedoch so weit verlangsamen möchten, dass der Werkzeugweg innerhalb von 0,05 Benutzereinheiten des programmierten Pfads bleibt. Die genaue Verlangsamung hängt von der Geometrie der programmierten Ecke und den Maschinenbeschränkungen ab, aber der Programmierer muss sich nur um die Toleranz kümmern. Dies gibt dem Programmierer vollständige Kontrolle über den Kompromiss beim Folgen eines Pfads. Die Blend-Toleranz kann im Programmverlauf bei Bedarf geändert werden. Beachten Sie, dass die Angabe von G64 P0 denselben Effekt hat wie G64 allein (siehe oben), was für die Rückwärtskompatibilität mit alten G-Code-Programmen notwendig ist. Siehe den G64-Abschnitt des G-Code-Kapitels.

Übergang (engl. blending) ohne Toleranz

Der kontrollierte Punkt berührt jede angegebene Bewegung an mindestens einem Punkt. Die Maschine wird sich nie mit einer solchen Geschwindigkeit bewegen, dass sie am Ende der aktuellen Bewegung (oder der nächsten Bewegung, wenn Sie eine Pause machen, wenn das Mischen bereits begonnen hat) nicht genau zum Stillstand kommen kann. Der Abstand vom Endpunkt der Bewegung ist so groß, wie er sein muss, um den besten Konturvorschub zu erhalten.

Naiver CAM-Detektor

Aufeinanderfolgende G1-Bewegungen, die nur die XYZ-Achsen betreffen und weniger als Q- von einer geraden Linie abweichen, werden zu einer einzigen geraden Linie zusammengeführt. Diese zusammengefasste Bewegung ersetzt die einzelnen G1-Bewegungen für die Zwecke der Überblendung mit Toleranz. Zwischen den aufeinanderfolgenden Bewegungen darf der kontrollierte Punkt nicht weiter als P- von den tatsächlichen Endpunkten der Bewegungen entfernt sein. Der kontrollierte Punkt berührt bei jeder Bewegung mindestens einen Punkt. Bei G2/3-Bewegungen in der G17 (XY)-Ebene, wenn die maximale Abweichung eines Bogens von einer geraden Linie kleiner ist als die G64 Q-Toleranz, wird der Bogen in zwei Linien unterteilt (vom Bogenanfang zum Mittelpunkt und vom Mittelpunkt zum Ende). Diese Linien unterliegen dann dem naiven Nockenalgorithmus für Linien. Auf diese Weise profitieren sowohl Linien-Bogen-, Bogen-Bogen- und Bogen-Linien-Fälle als auch Linien-Linien von dem "naive cam detector". Dies verbessert die Konturierungsleistung durch Vereinfachung des Pfades.

In der folgenden Abbildung stellt die blaue Linie die tatsächliche Maschinengeschwindigkeit dar. Die roten Linien stellen das Beschleunigungsvermögen der Maschine dar. Die horizontalen Linien unter jedem Diagramm stellen die geplante Bewegung dar. Das obere Diagramm zeigt, wie der Bahnplaner die Maschine bei kurzen Bewegungen abbremst, um innerhalb der Grenzen der Beschleunigungseinstellung der Maschine zu bleiben, damit sie am Ende der nächsten Bewegung exakt zum Stillstand kommt. Das untere Diagramm zeigt die Wirkung des naive cam detectors, der die Bewegungen kombiniert und die Geschwindigkeit besser als geplant beibehält.

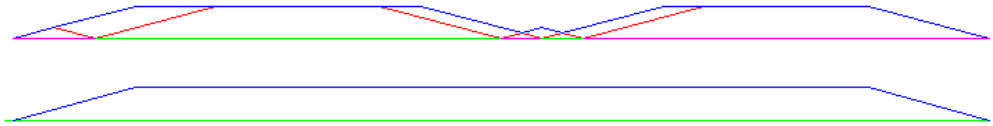


Figure 16. Naiver CAM-Detektor

Planen von Bewegungen

Vergewissern Sie sich, dass die Verfahrwege für Ihre Maschine/Ihr Material "lang genug" sind. Da sich die Maschine niemals mit einer solchen Geschwindigkeit bewegt, dass sie am Ende der aktuellen Bewegung nicht zum Stillstand kommen kann, gibt es eine Mindestbewegungslänge, die es der Maschine ermöglicht, die gewünschte Vorschubgeschwindigkeit bei einer bestimmten Beschleunigungseinstellung beizubehalten.

Die Beschleunigungs- und Abbremsphase verwenden jeweils die Hälfte der INI-Datei MAX_ACCELERATION. Bei einem Blend, der eine exakte Umkehrung darstellt, führt dies dazu, dass die Gesamtbeschleunigung der Achse gleich der MAX_ACCELERATION der INI-Datei ist. In anderen Fällen ist die tatsächliche Maschinenbeschleunigung etwas geringer als die Beschleunigung der INI-Datei.

Um den Vorschub aufrechtzuerhalten, muss die Bewegung länger sein als die Strecke, die benötigt wird, um von 0 auf die gewünschte Vorschubgeschwindigkeit zu beschleunigen und dann wieder anzuhalten. Wenn man A als **1/2** der INI-Datei MAX_ACCELERATION und F als Vorschubgeschwindigkeit **in Einheiten pro Sekunde** verwendet, ist die Beschleunigungszeit $\mathbf{t_a}$ = **F/A** und die Beschleunigungsstrecke $\mathbf{d_a}$ = **F*t$_a$/2**. Die Verzögerungszeit und -strecke sind gleich, so dass die kritische Strecke **d** = $\mathbf{d_a}$ + $\mathbf{d_d}$ = **2 *** $\mathbf{d_a}$ = $\mathbf{F^2/A}$ beträgt.

Bei einer Vorschubgeschwindigkeit von 1 Zoll pro Sekunde und einer Beschleunigung von **10 Zoll/sec^2** beträgt der kritische Abstand beispielsweise $\mathbf{1^2/10}$ = **1/10** = **0,1 Zoll**.

Bei einer Vorschubgeschwindigkeit von 0,5 Zoll pro Sekunde beträgt der kritische Abstand $\mathbf{5^2/100}$ = **25/100** = **0,025** Zoll.

2.3.2. G-Code

Standardeinstellungen

Beim ersten Start von LinuxCNC werden standardmäßig viele G- und M-Codes geladen. Die aktuellen aktiven G- und M-Codes können auf der Registerkarte MDI im Fenster *Aktive G-Codes:* in der AXIS-Schnittstelle angezeigt werden. Diese G- und M-Codes definieren das Verhalten von LinuxCNC und es ist wichtig, dass Sie verstehen, was jeder einzelne tut, bevor Sie LinuxCNC ausführen. Die Standardeinstellungen können geändert werden, wenn eine G-Code-Datei und links in einen anderen Zustand als wenn Sie Ihre LinuxCNC-Sitzung gestartet. Die beste Praxis ist es, die Vorgaben für den Job in der Präambel Ihrer G-Code-Datei benötigt und nicht davon ausgehen, dass die Vorgaben nicht geändert haben. Das Ausdrucken der G-Code Kurzübersicht Seite kann Ihnen helfen, sich zu erinnern, was jeder einzelne ist.

Vorschubgeschwindigkeit (engl. feed rate)

Wie die Vorschubgeschwindigkeit angewandt wird, hängt davon ab, ob eine an der Bewegung beteiligte Achse eine Drehachse ist. Lesen und verstehen Sie den Abschnitt Feed Rate, wenn Sie eine Rotationsachse (engl. rotary axis) oder eine Drehmaschine haben.

Werkzeugradius-Versatz

Der Werkzeugradius-Versatz (G41/42) setzt voraus, dass das Werkzeug in der Lage ist, jede programmierte Bewegung irgendwo zu berühren, ohne die beiden angrenzenden Bewegungen zu fugen. Wenn dies mit dem aktuellen Werkzeugdurchmesser nicht möglich ist, erhalten Sie eine Fehlermeldung. Ein Werkzeug mit kleinerem Durchmesser kann ohne Fehler auf demselben Weg laufen. Das bedeutet, dass Sie einen Fräser so programmieren können, dass er eine Bahn durchläuft, die schmaler ist als der Fräser, ohne dass ein Fehler auftritt. Weitere Informationen finden Sie im Abschnitt Cutter Compensation.

2.3.3. Referenzfahrt (engl. homing)

Nach dem Start von LinuxCNC muss jede Achse referenziert werden, bevor ein Programm oder ein MDI-Befehl ausgeführt wird. Wenn Ihre Maschine nicht über Referenzschalter verfügt, kann eine Übereinstimmungsmarkierung auf jeder Achse dabei helfen, die Maschinenkoordinaten jedes Mal auf die gleiche Position zu referenzieren. Nach der Referenzierung werden Ihre in der INI-Datei festgelegten Soft-Limits verwendet.

Wenn Sie vom Standardverhalten abweichen oder die Mini-Schnittstelle verwenden möchten, müssen Sie die Option `NO_FORCE_HOMING = 1` im Abschnitt `[TRAJ]` Ihrer INI-Datei setzen. Weitere Informationen zur Referenzfahrt finden Sie im Integrator-Handbuch.

2.3.4. Werkzeugwechsel

Bei manuellen Werkzeugwechseln gibt es mehrere Optionen. Siehe [EMCIO] Abschnitt für Informationen zur Konfiguration dieser Optionen. Siehe auch die Abschnitte G28 und G30 im Kapitel G-Code.

2.3.5. Koordinatensysteme

Die Koordinatensysteme können anfangs verwirrend sein. Bevor Sie eine CNC-Maschine betreiben, müssen Sie die Grundlagen der von LinuxCNC verwendeten Koordinatensysteme verstehen. Ausführliche Informationen über die LinuxCNC Koordinatensysteme finden Sie im Coordinate System Abschnitt dieses Handbuchs.

G53 Maschinenkoordinaten

Wenn Sie LinuxCNC referenzieren, setzen Sie das G53-Maschinenkoordinatensystem für jede referenzierte Achse auf 0.

Andere Koordinatensysteme oder Werkzeugversätze werden durch die Referenzfahrt nicht verändert.

Sie bewegen sich im G53-Maschinenkoordinatensystem nur, wenn Sie G53 auf der gleichen Linie wie eine Bewegung programmieren. Normalerweise befinden Sie sich im G54-Koordinatensystem.

G54-59.3 Benutzerkoordinaten

Normalerweise verwenden Sie das G54-Koordinatensystem. Wenn ein Offset auf ein aktuelles Benutzerkoordinatensystem angewendet wird, befindet sich eine kleine blaue Kugel mit Linien an der Maschinen-Ursprung (engl. origin) wenn Ihre DRO "Position: Relative Actual" in AXIS anzeigt. Wenn Ihre Offsets temporär sind, verwenden Sie das Nullkoordinatensystem aus dem Menü Maschine oder programmieren Sie *G10 L2 P1 X0 Y0 Z0* am Ende Ihrer G-Code-Datei. Ändern Sie die P-Nummer entsprechend dem Koordinatensystem, in dem Sie den Versatz löschen möchten.

- Offsets, die in einem Benutzerkoordinatensystem gespeichert sind, bleiben erhalten, wenn LinuxCNC heruntergefahren wird.
- Mit der Schaltfläche *Touch Off* in AXIS wird ein Offset für das gewählte Benutzerkoordinatensystem festgelegt.

Wenn Sie nicht mehr weiterwissen

Wenn Sie Probleme haben, (0,0,0) auf der DRO zu erreichen, wenn Sie denken, dass Sie es sollten, haben Sie vielleicht einige Offsets programmiert und müssen diese entfernen.

- Fahren Sie mit G53 G0 X0 Y0 Z0 zum Maschinennullpunkt
- Löschen Sie alle G92-Offsets mit G92.1
- Verwendung des G54-Koordinatensystems mit G54
- Stellen Sie das G54-Koordinatensystem mit `G10 L2 P1 X0 Y0 Z0 R0` auf das Maschinenkoordinatensystem ein.
- Werkzeugversätze mit G49 deaktivieren
- Aktivieren der relativen Koordinatenanzeige über das Menü

Jetzt sollten Sie sich am Maschinenursprung X0 Y0 Z0 befinden und das relative Koordinatensystem sollte mit dem Maschinenkoordinatensystem übereinstimmen.

2.3.6. Maschinenkonfigurationen

Das folgende Diagramm zeigt eine typische Fräsmaschine mit der Bewegungsrichtung des Werkzeugs, des Frästisches und der Endschalter. Beachten Sie, dass sich der Frästisch in die entgegengesetzte Richtung der Pfeile des kartesischen Koordinatensystems bewegt, die in der Abbildung "Werkzeugrichtung" dargestellt sind. Dadurch bewegt sich das "Werkzeug" in die richtige Richtung in Bezug auf das Material.

Beachten Sie auch die Position der Endschalter und die Richtung der Aktivierung ihrer Nocken. Es sind mehrere Kombinationen möglich, z. B. ist es möglich (entgegen der Zeichnung), einen einzigen festen Endschalter in der Mitte des Tisches zu platzieren und zwei bewegliche Nocken, um ihn zu aktivieren. In diesem Fall werden die Grenzen umgekehrt, +X befindet sich auf der rechten Seite des Tisches und -X auf der linken Seite. Diese Umkehrung ändert nichts an der Bewegungsrichtung des Werkzeugs.

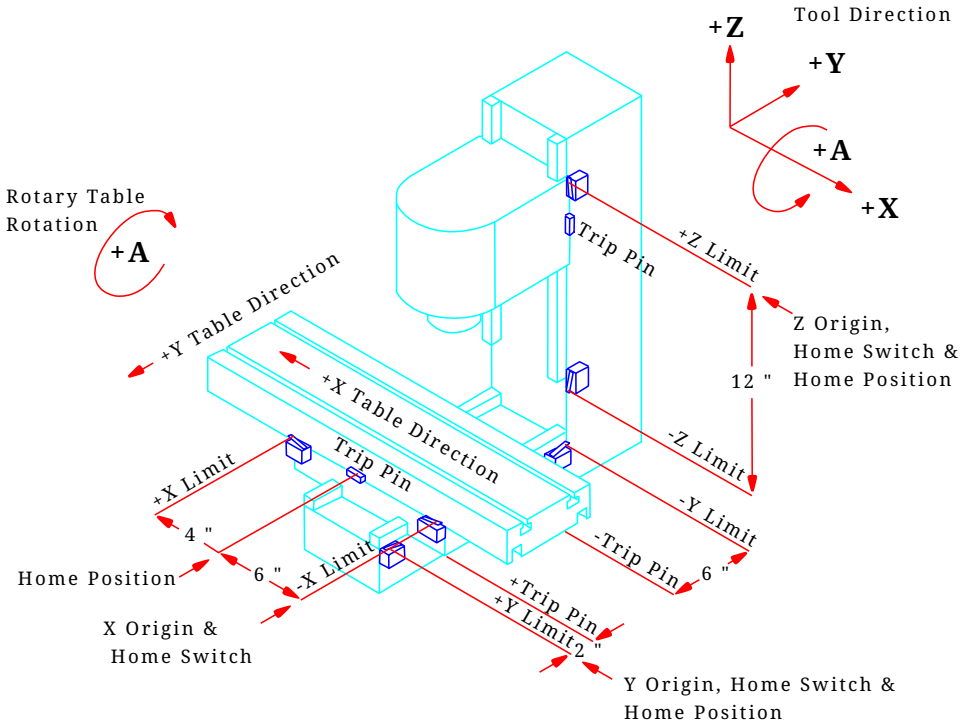


Figure 17. Typische Fräsen-Konfiguration

Die folgende Abbildung zeigt eine typische Drehmaschine mit Verfahrrichtung des Werkzeugs und Endschaltern.

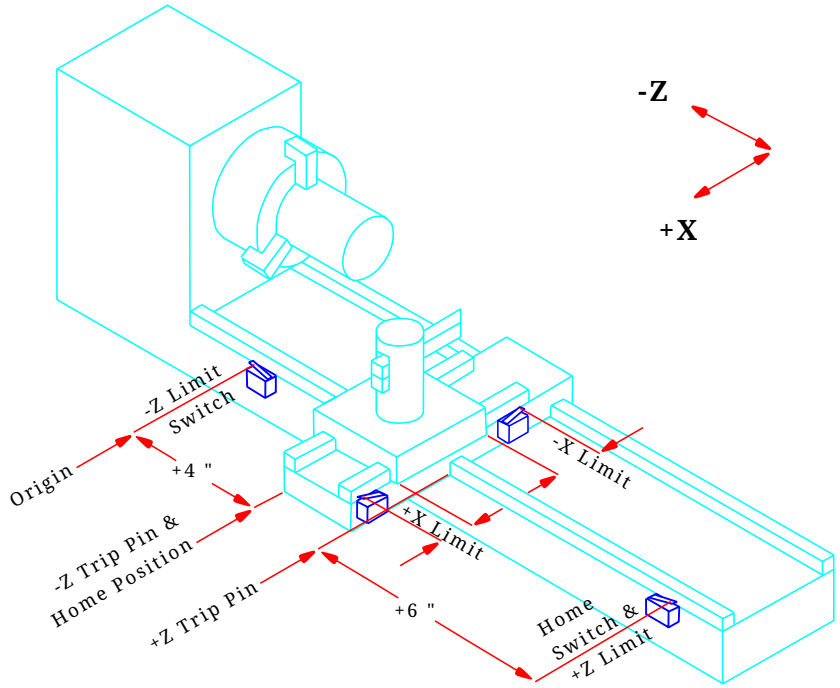


Figure 18. Typische Drehmaschinen-Konfiguration

2.4. LinuxCNC starten

2.4.1. LinuxCNC ausführen

LinuxCNC wird mit der Skriptdatei *linuxcnc* gestartet.

```
linuxcnc [options] [<INI-file>]
```

`linuxcnc` *Skript Optionen*

```
linuxcnc: Run LinuxCNC

Usage:
  $ linuxcnc -h
    This help

  $ linuxcnc [Optionen]
    Wählen Sie die Konfigurations-INI-Datei grafisch aus

  $ linuxcnc [Optionen] Pfad/zu/Ihrer_INI_Datei
    Benennen Sie die Konfigurations-INI-Datei anhand ihres Pfads

  $ linuxcnc [Options] -l
    Nutze die zuvor genutzte Konfigurations-INI-Datei

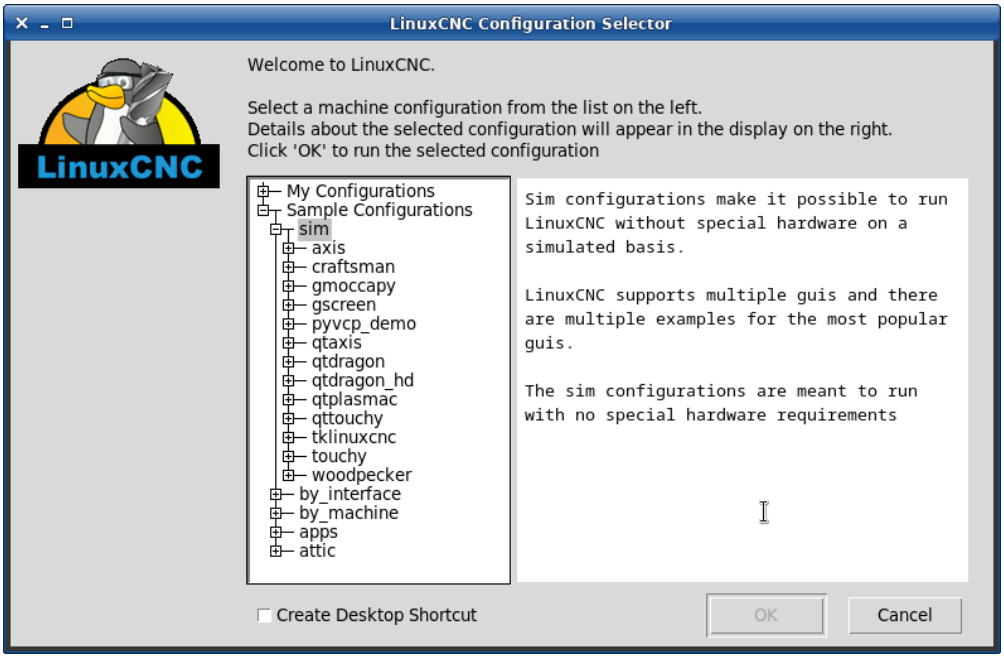
Options:
    -d: Turn on "debug" mode
    -v: Turn on "verbose" mode
    -r: Disable redirection of stdout and stderr to ~/linuxcnc_print.txt and
        ~/linuxcnc_debug.txt when stdin is not a tty.
        Used when running linuxcnc tests non-interactively.
    -l: Use the last-used INI file
    -k: Continue in the presence of errors in HAL files
    -t "tpmodulename [parameters]"
            specify custom trajectory_planning_module
            overrides optional INI setting [TRAJ]TPMOD
    -m "homemodulename [parameters]"
            specify custom homing_module
            overrides optional INI setting [EMCMOT]HOMEMOD
    -H "dirname": search dirname for HAL files before searching
                  INI directory and system library:
                  /home/git/linuxcnc-dev/lib/hallib
Note:
    The -H "dirname" option may be specified multiple times
```

Wenn das linuxcnc Skript eine INI-Datei übergeben wird, liest es die INI-Datei und startet LinuxCNC. Der Abschnitt der INI-Datei [HAL] gibt die Reihenfolge des Ladens der HAL-Dateien an, wenn mehr als eine verwendet wird. Sobald die HAL=xxx.hal Dateien geladen sind, wird die GUI geladen und dann die POSTGUI=.xxx.hal Datei. Wenn Sie PyVCP- oder GladeVCP-Objekte mit HAL-Pins erstellen, müssen Sie die postgui-HAL-Datei verwenden, um Verbindungen zu diesen Pins herzustellen. Siehe den Abschnitt [HAL] der INI-Konfiguration für weitere Informationen.

Konfigurationsauswahl

Wenn keine INI-Datei an das linuxcnc Skript übergeben wird, lädt es den Konfigurationsselektor, so dass

Sie eine Beispielkonfiguration auswählen und speichern können. Sobald eine Beispielkonfiguration gespeichert wurde, kann sie an die eigene Anwendung angepasst werden. Die Konfigurationsdateien werden im Verzeichnis linuxcnc/configs gespeichert.



2.5. CNC-Maschinenübersicht

In diesem Abschnitt wird kurz beschrieben, wie eine CNC-Maschine von der Eingangs- und Ausgangsseite des Interpreters aus betrachtet wird.

2.5.1. Mechanische Bestandteile

Eine CNC-Maschine hat viele mechanische Komponenten, die gesteuert werden können oder die Art und Weise der Steuerung beeinflussen können. Dieser Abschnitt beschreibt die Teilmenge dieser Komponenten, die mit dem Interpreter interagieren. Mechanische Komponenten, die nicht direkt mit dem Interpreter interagieren, wie z. B. die Jog Buttons/ Tipptasten, werden hier nicht beschrieben, auch wenn sie die Steuerung beeinflussen.

Achsen

Jede CNC-Maschine hat eine oder mehrere Achsen. Verschiedene Arten von CNC-Maschinen haben unterschiedliche Kombinationen. Eine "4-Achsen-Fräsmaschine" kann zum Beispiel XYZA- oder XYZB-Achsen haben. Eine Drehmaschine hat normalerweise XZ-Achsen. Eine Schaumstoffschneidemaschine kann XYUV-Achsen haben. In LinuxCNC, der Fall eines XYYZ "Gantry"-Maschine mit zwei Motoren für eine Achse ist besser durch Kinematik als durch eine zweite lineare Achse behandelt.

NOTE Wenn die Bewegung der mechanischen Komponenten nicht unabhängig ist, wie z. B. bei Hexapod-Maschinen, können die RS274/NGC-Sprache und die kanonischen Bearbeitungsfunktionen immer noch verwendet werden, solange die unteren Steuerungsebenen wissen, wie die tatsächlichen Mechanismen zu steuern sind, um die gleiche relative Bewegung von Werkzeug und Werkstück zu erzeugen, wie sie von

unabhängigen Achsen erzeugt würde. Dies wird als "Kinematik" bezeichnet.

NOTE Mit LinuxCNC, der für den Fall der XYYZ-Portal-Maschine mit zwei Motoren für eine Achse ist besser durch die Kinematik als durch eine zusätzliche lineare Achse behandelt.

.Primary Linear Axes The X, Y, and Z axes produce linear motion in three mutually orthogonal directions.

.Secondary Linear Axes The U, V, and W axes produce linear motion in three mutually orthogonal directions. Typically, X and U are parallel, Y and V are parallel, and Z and W are parallel.

.Rotational Axes The A, B and C axes produce angular motion (rotation). Typically, A rotates around a line parallel to X, B rotates around a line parallel to Y, and C rotates around a line parallel to Z.

Spindel

Eine CNC-Maschine hat in der Regel eine Spindel, die ein Zerspanungswerkzeug, einen Messtaster oder im Falle einer Drehmaschine das Material hält. Die Spindel kann, muss aber nicht von der CNC-Software gesteuert werden. LinuxCNC bietet Unterstützung für bis zu 8 Spindeln, die individuell gesteuert werden können und gleichzeitig mit unterschiedlichen Geschwindigkeiten und in unterschiedlichen Richtungen laufen können.

Kühlmittel

Flutkühlmittel und Nebelkühlmittel können unabhängig voneinander eingeschaltet werden. Die Sprache RS274/NGC schaltet sie gemeinsam aus, siehe Abschnitt M7 M8 M9.

Vorschub- und Geschwindigkeits-Neufestsetzungen (engl. overrides)

Eine CNC-Maschine kann über separate Vorschub- und Geschwindigkeitssteuerungen verfügen, mit denen der Bediener festlegen kann, dass der tatsächliche Vorschub oder die Spindeldrehzahl bei der Bearbeitung einen bestimmten Prozentsatz der programmierten Geschwindigkeit beträgt.

Schalter zum Löschen von Blöcken

Eine CNC-Maschine kann einen Schalter zum Löschen von Blöcken haben. Siehe den Abschnitt Block-Lösch-Schalter (engl. block delete switch).

Optionaler Programm-Stopp-Schalter

Eine CNC-Maschine kann mit einem optionalen Programmstoppschalter ausgestattet sein. Siehe den Abschnitt Optionaler Programmstopp.

2.5.2. Steuerungs- und Datenkomponenten

Lineare Achsen

Die X-, Y- und Z-Achse bilden ein standardmäßiges rechtshändiges Koordinatensystem mit orthogonalen linearen Achsen. Die Positionen der drei linearen Bewegungsmechanismen werden durch Koordinaten auf diesen Achsen ausgedrückt.

Die Achsen U, V und W bilden ebenfalls ein standardmäßiges rechtshändiges Koordinatensystem. X und U sind parallel, Y und V sind parallel, und Z und W sind parallel (wenn A, B und C auf Null gedreht werden).

Rotationsachsen

Die Rotationsachsen werden in Grad als umschlungene lineare Achsen gemessen, bei denen die positive Drehrichtung vom positiven Ende der entsprechenden X-, Y- oder Z-Achse aus gesehen gegen den Uhrzeigersinn ist. Mit "umschlungener Linearachse" ist eine Achse gemeint, bei der die Winkelposition unbegrenzt zunimmt (gegen plus unendlich geht), wenn sich die Achse gegen den Uhrzeigersinn dreht, und unbegrenzt abnimmt (gegen minus unendlich geht), wenn sich die Achse im Uhrzeigersinn dreht. Umschlungene lineare Achsen werden unabhängig davon verwendet, ob es eine mechanische Begrenzung der Drehung gibt oder nicht.

Im Uhrzeigersinn oder gegen den Uhrzeigersinn ist vom Standpunkt des Werkstücks aus gesehen. Wenn das Werkstück an einem Drehtisch befestigt ist, der sich um eine Drehachse dreht, wird eine Drehung gegen den Uhrzeigersinn aus Sicht des Werkstücks dadurch erreicht, dass der Drehtisch in eine Richtung gedreht wird, die (bei den meisten gängigen Maschinenkonfigurationen) aus Sicht einer neben der Maschine stehenden Person im Uhrzeigersinn aussieht. Fußnote:[Wenn die Anforderung der Parallelität verletzt wird, muss der Systemersteller angeben, wie zwischen Uhrzeigersinn und Gegenuhrzeigersinn unterschieden wird.]

Kontrollpunkt

Der gesteuerte Punkt ist der Punkt, dessen Position und Bewegungsgeschwindigkeit gesteuert werden. Wenn der Werkzeuglängenversatz Null ist (der Standardwert), ist dies ein Punkt auf der Spindelachse (häufig als Messpunkt bezeichnet), der sich in einem festen Abstand hinter dem Ende der Spindel befindet, normalerweise nahe dem Ende eines passenden Werkzeughalters in die Spindel. Die Position des gesteuerten Punkts kann entlang der Spindelachse verschoben werden, indem ein positiver Betrag für den Werkzeuglängenversatz angegeben wird. Dieser Betrag ist normalerweise die Länge des verwendeten Schneidwerkzeugs, so dass der kontrollierte Punkt am Ende des Schneidwerkzeugs liegt. Auf einer Drehmaschine können Werkzeuglängen-Offsets für die X- und Z-Achse angegeben werden, und der kontrollierte Punkt befindet sich entweder an der Werkzeugspitze oder etwas außerhalb davon (wo die senkrechten, achsenausgerichteten Linien von der *Vorderseite* und *Seite* von berührt werden das Werkzeug schneiden).

Koordinierte lineare Bewegung

Um ein Werkzeug entlang einer bestimmten Bahn zu bewegen, muss ein Bearbeitungszentrum häufig die Bewegung mehrerer Achsen koordinieren. Wir verwenden den Begriff "koordinierte lineare Bewegung", um die Situation zu beschreiben, in der sich nominell jede Achse mit konstanter Geschwindigkeit bewegt und sich alle Achsen gleichzeitig von ihren Startpositionen zu ihren

Endpositionen bewegen. Wenn sich nur die X-, Y- und Z-Achse (oder eine oder zwei von ihnen) bewegen, führt dies zu einer geradlinigen Bewegung, daher das Wort "linear" in dem Begriff. Bei tatsächlichen Bewegungen ist es oft nicht möglich, eine konstante Geschwindigkeit beizubehalten, da am Anfang und/oder am Ende der Bewegung eine Beschleunigung oder Verzögerung erforderlich ist. Es ist jedoch möglich, die Achsen so zu steuern, dass jede Achse zu jedem Zeitpunkt den gleichen Teil ihrer erforderlichen Bewegung ausgeführt hat wie die anderen Achsen. Dadurch wird das Werkzeug auf demselben Weg bewegt, und wir nennen diese Art der Bewegung auch koordinierte lineare Bewegung.

Koordinierte lineare Bewegungen können entweder mit der vorherrschenden Vorschubgeschwindigkeit oder mit der Verfahrgeschwindigkeit ausgeführt werden oder sie können mit der Spindelrotation synchronisiert werden. Wenn die gewünschte Geschwindigkeit aufgrund physikalischer Grenzen der Achsengeschwindigkeit nicht erreicht werden kann, werden alle Achsen verlangsamt, um die gewünschte Bahn beizubehalten.

Vorschubgeschwindigkeit (engl. feed rate)

Die Geschwindigkeit, mit der sich der gesteuerte Punkt bewegt, ist nominell eine stetige Geschwindigkeit, die vom Benutzer eingestellt werden kann. Im Interpreter wird die Vorschubgeschwindigkeit wie folgt interpretiert (es sei denn, die Modi "inverser Zeitvorschub" oder "Vorschub pro Umdrehung" werden verwendet; in diesem Fall siehe Abschnitt G93-G94-G95-Mode,G93 G94 G95).

1. If any of the feed axes are moving, F is in units per minute in the cartesian system of the feed axes, and all other axes move so as to start and stop in coordinated fashion. The feed axes are X Y Z unless `[TRAJ]FEED_AXES` in the INI file names others.
2. Otherwise, if any of the other linear axes are moving, F is in units per minute in their cartesian system, and the angular axes move so as to start and stop in coordinated fashion. These are U V W unless `[AXIS_<letter>]TYPE` in the INI file says otherwise.
3. Otherwise, the move is pure rotary motion and the F word is in rotary units in the *pseudo-cartesian* system of the angular axes, A B C unless `[AXIS_<letter>]TYPE` says otherwise.

Kühlen

Die Kühlung von Flut- oder Tröpfchen kann separat aktiviert werden. RS274 / NGC-Sprache stoppt sie zusammen. Siehe Abschnitt über Kühlsteuerung.

Verweilen (engl. dwell)

Ein Bearbeitungszentrum kann angewiesen werden, für eine bestimmte Zeit zu verweilen (d.h. alle Achsen unbeweglich zu halten). Die häufigste Anwendung des Verweilens ist das Brechen und Entfernen von Spänen, so dass sich die Spindel während des Verweilens normalerweise dreht. Unabhängig vom Bahnsteuerungsmodus (siehe Abschnitt Path Control) hält die Maschine genau am Ende der vorhergehenden programmierten Bewegung an, so als ob sie im exakten Bahnmodus wäre.

Einheiten

Units used for distances along the X, Y, and Z axes, and along any other linear axis, may be measured in millimeters or inches. Units for all other quantities involved in machine control cannot be changed. Different quantities use different specific units. Spindle speed is measured in revolutions per minute. The positions of rotational axes are measured in degrees. An axis is linear or rotational as `[AXIS_<letter>]TYPE` in the INI file says: U V W linear and A B C rotational unless it says otherwise. Feed rates are expressed in current length units per minute, or degrees per minute, or length units per spindle revolution, as described in section G93 G94 G95.

Aktuelle Position

Der kontrollierte Punkt befindet sich immer an einer Stelle, die als "aktuelle Position" bezeichnet wird, und der Controller weiß immer, wo sich diese befindet. Die dargestellte aktuelle Position muss angepasst werden, selbst wenn keine Achsenbewegung stattfindet, wenn eines von mehreren Ereignissen eintritt:

1. Längeneinheiten werden geändert.
2. Werkzeuglängenkorrektur wird geändert.
3. Koordinatensystem-Offsets werden geändert.

Ausgewählte Ebene

Es gibt immer eine *ausgewählte Ebene*, welche die XY-Ebene, die YZ-Ebene oder die XZ-Ebene des Bearbeitungszentrums sein muss. Die Z-Achse steht natürlich senkrecht auf der XY-Ebene, die X-Achse auf der YZ-Ebene und die Y-Achse auf der XZ-Ebene.

Werkzeug-Karussell

Jedem Steckplatz im Werkzeugkarussell wird ein oder null Werkzeuge zugewiesen.

Werkzeugwechsel

Einem Bearbeitungszentrum kann der Befehl zum Werkzeugwechsel gegeben werden.

Paletten-Shuttle

Die beiden Paletten können auf Befehl ausgetauscht werden.

Geschwindigkeits-Neufestsetzung (engl. override)

Die Tasten für die Geschwindigkeits-Neufestsetzung können aktiviert (sie funktionieren normal) oder deaktiviert werden (sie haben keine Wirkung mehr). In der Sprache RS274/NGC gibt es einen Befehl, der alle Tasten aktiviert, und einen anderen, der sie deaktiviert. Siehe Sperrung und Aktivierung Geschwindigkeits-Korrektur. Siehe auch hier für weitere Details.

Pfadsteuerungsmodus

Das Bearbeitungszentrum kann in einen von drei Bahnsteuerungsmodi versetzt werden:

exakter Stoppmodus

Im Exaktstopp-Modus hält die Maschine am Ende jeder programmierten Bewegung kurz an.

genauer Pfadmodus (engl. exact path mode)

Im exakten Pfadmodus folgt die Maschine dem programmierten Weg so genau wie möglich und verlangsamt oder stoppt bei Bedarf an scharfen Ecken des Weges.

kontinuierlicher Modus (continuous mode)

Im kontinuierlichen Modus können scharfe Ecken der Bahn leicht abgerundet werden, damit die Vorschubgeschwindigkeit beibehalten werden kann (aber ohne Verletzung der Toleranzgrenzen, falls angegeben).

Siehe Abschnitte G61 und G64.

2.5.3. Interpreter-Interaktion mit Schaltern

Der Interpreter interagiert mit mehreren Schaltern. In diesem Abschnitt werden die Interaktionen genauer beschrieben. In keinem Fall weiß der Interpreter, wie die Einstellung eines dieser Schalter ist.

Schalter für Vorschub- und Geschwindigkeits-Neufestsetzungen (engl. overrides)

Der Interpreter interpretiert RS274/NGC-Befehle zur Aktivierung (*M48*) oder Deaktivierung (*M49*) des Vorschub- und Geschwindigkeitsüberbrückungsschalters. Für bestimmte Bewegungen, wie z.B. das Herausfahren aus dem Ende eines Gewindes während eines Einfädelzyklus, werden die Schalter automatisch deaktiviert.

LinuxCNC reagiert auf die Geschwindigkeit und Vorschub Neufestsetzungs (engl. override)-Einstellungen, wenn diese Schalter aktiviert sind.

Siehe den Abschnitt zu M48 M49 Neufestsetzungen für weitere Informationen.

Schalter zum Löschen von Blöcken

Wenn der Schalter für das Löschen von Blöcken aktiviert ist, werden G-Code-Zeilen, die mit einem Schrägstrich (dem Blocklöschzeichen) beginnen, nicht interpretiert. Wenn der Schalter ausgeschaltet ist, werden solche Zeilen interpretiert. Normalerweise sollte der Blocklöschschalter vor dem Start des NGC-Programms gesetzt werden.

Optionaler Programm-Stopp-Schalter

Wenn dieser Schalter eingeschaltet ist und ein M1-Code auftritt, wird die Programmausführung angehalten.

2.5.4. Werkzeugtabelle

Für die Verwendung des Interpreters ist eine Werkzeugtabelle erforderlich. In dieser Datei ist festgelegt, welche Werkzeuge sich in welchen Werkzeugwechslerplätzen befinden und welche Größe und welchen Typ die einzelnen Werkzeuge haben. Der Name der Werkzeugtabelle wird in der INI-Datei definiert:

```
[EMCIO]
# Datei mit Werkzeugtabelle
TOOL_TABLE = tooltable.tbl
```

Der Standard-Dateiname sieht wahrscheinlich so aus wie oben, aber Sie können es vorziehen, Ihrer Maschine eine eigene Werkzeugtabelle zu geben, die denselben Namen wie Ihre INI-Datei trägt, aber mit der Erweiterung tbl:

```
TOOL_TABLE = acme_300.tbl
```

oder:

```
TOOL_TABLE = EMC-AXIS-SIM.tbl
```

Weitere Informationen zu den Besonderheiten des Formats der Werkzeugtabelle finden Sie im Abschnitt Werkzeugtabellen-Format.

2.5.5. Parameter

In der RS274/NGC-Sprachansicht verwaltet ein Bearbeitungszentrum eine Reihe von numerischen Parametern, die durch eine Systemdefinition (RS274NGC_MAX_PARAMETERS) festgelegt sind. Viele von ihnen haben spezifische Verwendungen, insbesondere bei der Definition von Koordinatensystemen. Die Anzahl der numerischen Parameter kann sich erhöhen, wenn die Entwicklung die Unterstützung für neue Parameter hinzufügt. Das Parameter-Array bleibt über die Zeit erhalten, auch wenn das Bearbeitungszentrum ausgeschaltet ist. LinuxCNC verwendet eine Parameterdatei, um die Persistenz zu gewährleisten und gibt dem Interpreter die Verantwortung für die Pflege der Datei. Der Interpreter liest die Datei, wenn er startet, und schreibt die Datei, wenn er beendet wird.

Alle Parameter sind für die Verwendung in G-Code-Programmen verfügbar.

Das Format einer Parameterdatei ist in der folgenden Tabelle dargestellt. Die Datei besteht aus einer beliebigen Anzahl von Kopfzeilen, gefolgt von einer Leerzeile, gefolgt von einer beliebigen Anzahl von Datenzeilen. Der Interpreter überspringt die Kopfzeilen. Wichtig ist, dass vor den Daten genau eine Leerzeile (auch ohne Leerzeichen oder Tabulatoren) steht. Die in der folgenden Tabelle gezeigte Kopfzeile beschreibt die Datenspalten. Es wird daher vorgeschlagen (ist aber nicht erforderlich), diese Zeile immer in die Kopfzeile aufzunehmen.

Der Interpreter liest nur die ersten beiden Spalten der Tabelle. Die dritte Spalte, *Kommentar* (engl. comment), wird vom Interpreter nicht gelesen.

Jede Zeile der Datei enthält in der ersten Spalte die Indexnummer eines Parameters und in der zweiten Spalte den Wert, auf den dieser Parameter gesetzt werden soll. Der Wert wird im Interpreter als

Gleitkommazahl mit doppelter Genauigkeit dargestellt, aber ein Dezimalpunkt ist in der Datei nicht erforderlich. Alle in der folgenden Tabelle aufgeführten Parameter sind obligatorisch und müssen in jeder Parameterdatei enthalten sein, mit der Ausnahme, dass jeder Parameter, der einen Drehachsenwert für eine nicht verwendete Achse darstellt, weggelassen werden kann. Wenn ein erforderlicher Parameter fehlt, wird ein Fehler gemeldet. Eine Parameterdatei kann jeden anderen Parameter enthalten, solange seine Nummer im Bereich von 1 bis 5400 liegt. Die Parameternummern müssen in aufsteigender Reihenfolge angeordnet sein. Ist dies nicht der Fall, wird ein Fehler gemeldet. Jeder Parameter, der in der vom Interpreter gelesenen Datei enthalten ist, wird auch in die Datei aufgenommen, die er beim Beenden des Programms schreibt. Die Originaldatei wird beim Schreiben der neuen Datei als Sicherungsdatei gespeichert. Kommentare werden beim Schreiben der Datei nicht beibehalten.

Table 2. Parameter-Dateiformat

Parameter-Nummer	Parameter-Wert	Kommentar
5161	0.0	G28 Referenzfahrt (engl. home) X
5162	0.0	G28 Referenzfahrt (engl. home)

Siehe den Abschnitt zu Parametern für weitere Informationen.

2.6. Drehmaschinen Anwender-Information

Dieses Kapitel enthält Informationen speziell für Drehmaschinen.

2.6.1. Drehbank-Modus

Wenn Ihre CNC-Maschine eine Drehmaschine ist, gibt es einige spezifische Änderungen, die Sie wahrscheinlich an Ihrer INI-Datei vornehmen möchten, um die besten Ergebnisse von LinuxCNC zu erzielen.

Wenn Sie das AXIS-Display verwenden, müssen Sie dafür sorgen, dass AXIS Ihre Drehwerkzeuge richtig anzeigt. Siehe den Abschnitt INI Konfiguration für weitere Details.

Erstellung von AXIS für Drehmaschinen-Modus.

```
[DISPLAY]

# Teilen Sie dem AXIS GUI mit, dass unsere Maschine eine Drehmaschine ist.
LATHE = TRUE
```

Der Drehmaschinenmodus in AXIS setzt Ihre Standardebene nicht auf G18 (XZ). Sie müssen dies in der Präambel jeder G-Code-Datei programmieren oder (besser) in Ihre INI-Datei aufnehmen, etwa so:

```
[RS274NGC]
```

```
# G-Code Modalcodes (Modi), mit denen der Interpreter initialisiert wird
# beim Starten
RS274NGC_STARTUP_CODE = G18 G20 G90
```

Wenn Sie GMOCCAPY verwenden, lesen Sie den GMOCCAPY-Drehmaschine-Abschnitt.

2.6.2. Drehmaschinen Werkzeugtabelle

Die "Werkzeugtabelle" ist eine Textdatei, die Informationen über jedes Werkzeug enthält. Die Datei befindet sich in demselben Verzeichnis wie Ihre Konfiguration und heißt standardmäßig "tool.tbl". Die Werkzeuge können sich in einem Werkzeugwechsler befinden oder einfach manuell geändert werden. Die Datei kann mit einem Texteditor bearbeitet oder mit G10 L1,L10,L11 aktualisiert werden. Es gibt auch einen eingebauten Werkzeugtabelleneditor in dem AXIS GUI. Die maximale Anzahl der Einträge in der Werkzeugtabelle beträgt 56. Die maximale Anzahl von Werkzeugen und Plätzen beträgt 99999.

Frühere Versionen von LinuxCNC hatten zwei verschiedene Werkzeugtabellenformate für Fräse n und Drehmaschinen, aber seit der Version 2.4.x wird ein Werkzeugtabellenformat für alle Maschinen verwendet. Ignorieren Sie einfach die Teile der Werkzeugtabelle, die nicht zu Ihrer Maschine gehören oder die Sie nicht benötigen. Weitere Informationen zu den Besonderheiten des Werkzeugtabellenformats finden Sie im Abschnitt Werkzeug-Table.

2.6.3. Drehmaschinen Werkzeug-Ausrichtung

Die folgende Abbildung zeigt die Ausrichtungen der Drehmeißel mit dem Winkel der Mittellinie jeder Ausrichtung und Informationen zu VORDERWINKEL und HINTERWINKEL.

FRONTANGLE und BACKANGLE sind im Uhrzeigersinn beginnend an einer Linie parallel zu Z+.

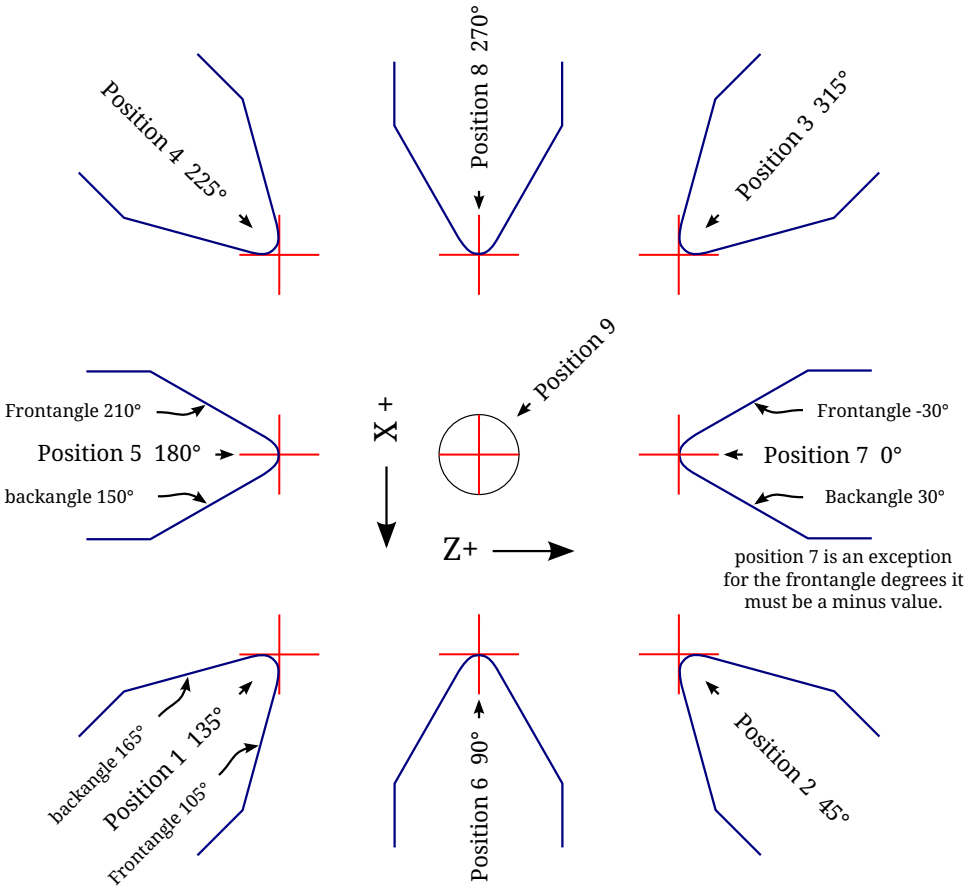
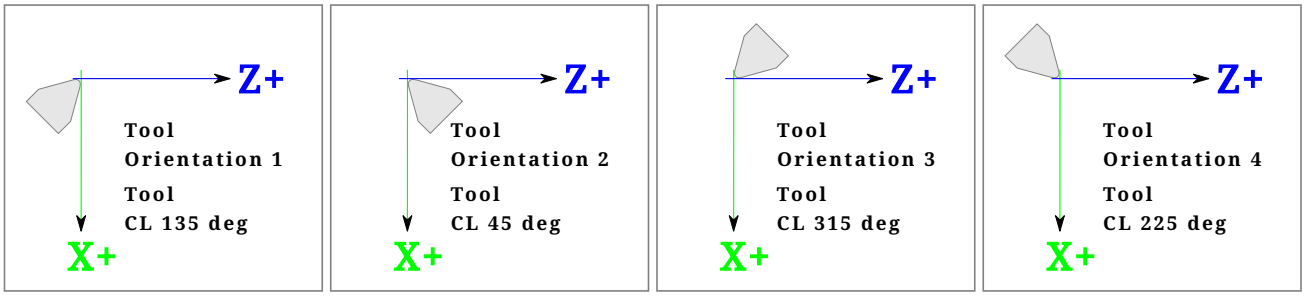


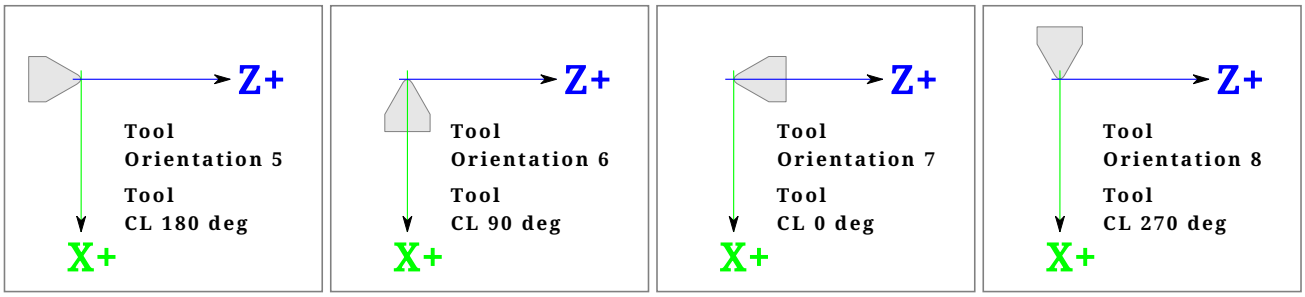
Figure 19. Ausrichtung des Drehwerkzeugs

In AXIS zeigen die folgenden Abbildungen, wie die Werkzeugpositionen aussehen, wie sie in der Werkzeugtabelle eingegeben wurden.

Werkzeugpositionen 1, 2, 3 & 4



Werkzeugpositionen 5, 6, 7 & 8



2.6.4. Werkzeug Touch Off

Wenn Sie in AXIS im Drehmaschinenmodus arbeiten, können Sie die X- und Z-Werte in der Werkzeugtabelle mit Hilfe des Touch Off-Fensters einstellen. Wenn Sie über einen Werkzeugrevolver verfügen, haben Sie beim Einrichten des Revolvers normalerweise die Option "Anlegen an die Vorrichtung" ausgewählt. Beim Einstellen des Z-Nullpunkts für das Material ist die Option "Auf das Material aufsetzen" ausgewählt. Weitere Informationen über die für Werkzeuge verwendeten G-Codes finden Sie unter M6, T*n* und G43. Für weitere Informationen zu den Optionen für die Werkzeugberührung in AXIS siehe Tool Touch Off.

X Touch Off

Der X-Achsen-Versatz für jedes Werkzeug ist normalerweise ein Versatz von der Mittellinie der Spindel.

Eine Methode besteht darin, Ihr normales Drehwerkzeug zu nehmen und ein Rohteil mit einem bekannten Durchmesser abzudrehen. Geben Sie den gemessenen Durchmesser (oder den Radius, wenn Sie sich im Radiusmodus befinden) für dieses Werkzeug in das Fenster "Tool Touch Off" ein. Verwenden Sie dann etwas Layout-Fluid oder einen Marker, um das Teil zu beschichten, bringen Sie jedes Werkzeug nach oben, bis es die Farbe gerade berührt, und stellen Sie seinen X-Offset auf den Durchmesser des Teils ein, das Sie mit dem Werkzeug-Anlegefenster verwenden. Stellen Sie sicher, dass bei allen Werkzeugen in den Eckquadranten der Nasenradius in der Werkzeugtabelle richtig eingestellt ist, damit der Kontrollpunkt korrekt ist. Beim Ansetzen des Werkzeugs wird automatisch ein G43 hinzugefügt, so dass das aktuelle Werkzeug den aktuellen Versatz darstellt.

Eine typische Sitzung könnte so aussehen:

1. Jede Achse referenzieren (engl. home), wenn sie nicht ausgerichtet ist.
2. Setzen Sie das aktuelle Werkzeug mit *Tn M6 G43*, wobei *n* die Werkzeugnummer ist.
3. Wählen Sie die X-Achse im Fenster "Manuelle Steuerung".
4. Bringen Sie das X in eine bekannte Position oder machen Sie einen Testschnitt und messen Sie den Durchmesser.
5. Wählen Sie Touch Off und wählen Sie Werkzeugtabelle, dann geben Sie die Position oder den Durchmesser ein.
6. Gehen Sie in der gleichen Reihenfolge vor, um die Z-Achse zu korrigieren.

Hinweis: Wenn Sie sich im Radiusmodus befinden, müssen Sie den Radius und nicht den Durchmesser eingeben.

Z Touch-Off

Die Versätze der Z-Achse können anfangs etwas verwirrend sein, da der Z-Versatz aus zwei Elementen besteht. Es gibt den Werkzeugtischversatz und den Maschinenkoordinatenversatz. Zunächst werden wir uns mit den Versätzen auf dem Werkzeugtisch befassen. Eine Methode besteht darin, einen festen Punkt auf Ihrer Drehmaschine zu verwenden und den Z-Versatz für alle Werkzeuge von diesem Punkt aus einzustellen. Manche verwenden die Spindelnase oder die Futterfläche. Auf diese Weise können Sie zu einem neuen Werkzeug wechseln und dessen Z-Versatz einstellen, ohne alle Werkzeuge neu einstellen zu müssen.

Eine typische Sitzung könnte so aussehen:

1. Jede Achse referenzieren (engl. home), wenn sie nicht ausgerichtet ist.
2. Vergewissern Sie sich, dass für das aktuelle Koordinatensystem keine Offsets gelten.
3. Setzen Sie das aktuelle Werkzeug mit *Tn M6 G43*, wobei *n* die Werkzeugnummer ist.
4. Wählen Sie die Z-Achse im Fenster Manuelle Steuerung.
5. Bringen Sie das Werkzeug nahe an die Steuerfläche.
6. Bewegen Sie das Z mit einem Zylinder von der Steuerfläche weg, bis der Zylinder gerade zwischen dem Werkzeug und der Steuerfläche durchläuft.
7. Wählen Sie Berühren (engl. touch off) und wählen Sie Werkzeugtabelle und setzen Sie die Position auf 0.0.
8. Wiederholen Sie diesen Vorgang für jedes Werkzeug mit demselben Zylinder.

Jetzt sind alle Werkzeuge um den gleichen Abstand von einer Standardposition versetzt. Wenn Sie ein Werkzeug, z. B. eine Bohrkrone, wechseln, wiederholen Sie die obigen Schritte und das Werkzeug ist nun mit den anderen Werkzeugen für den Z-Versatz synchronisiert. Bei einigen Werkzeugen ist es erforderlich, den Kontrollpunkt vom Aufsetzpunkt aus zu bestimmen. Wenn Sie z. B. ein 0,125 Zoll breites Abstechwerkzeug haben und die linke Seite abtasten, die rechte Seite aber Z0 sein soll, dann geben Sie 0.125 Zoll in das Absetzfenster ein.

Der Z-Maschinenversatz (engl. machine offset)

Sobald für alle Werkzeuge die Z-Korrektur in die Werkzeugtabelle eingegeben wurde, können Sie mit jedem Werkzeug die Maschinen-Korrektur über das Maschinenkoordinatensystem einstellen.

Eine typische Sitzung könnte so aussehen:

1. Jede Achse referenzieren (engl. home), wenn sie nicht ausgerichtet ist.
2. Stellen Sie das aktuelle Werkzeug mit *Tn M6* ein, wobei *n* die Werkzeugnummer ist.
3. Geben Sie ein G43 aus, damit die aktuelle Werkzeugkorrektur wirksam wird.
4. Führen Sie das Werkzeug an das Werkstück heran und stellen Sie den Z-Offset der Maschine ein.

Wenn Sie vergessen, den G43 für das aktuelle Werkzeug einzustellen, wenn Sie den Versatz des Maschinenkoordinatensystems festlegen, erhalten Sie nicht das, was Sie erwarten, da der Werkzeugversatz zum aktuellen Versatz addiert wird, wenn das Werkzeug in Ihrem Programm

verwendet wird.

2.6.5. Spindelsynchronisierte Bewegung

Die spindel-synchronisierte Bewegung erfordert einen an die Spindel angeschlossenen Quadratur-Encoder mit einem Indeximpuls pro Umdrehung. Weitere Informationen finden Sie in der Motion-Manpage und im Spindel Steuerungs-Beispiel.

Gewinde-Drehen (engl. threading)

Der Gewindeschneidzyklus G76 wird sowohl für Innen- als auch für Außengewinde verwendet. Weitere Informationen finden Sie im Abschnitt G76.

Konstante Oberflächengeschwindigkeit

Konstante Oberflächenberechnung (engl. kurz CSS für Constant Surface Speed) verwendet den X-Ursprung der Maschine, modifiziert durch den X-Versatz des Werkzeugs, um die Spindeldrehzahl in U/min zu berechnen. CSS verfolgt Änderungen der Werkzeugkorrekturen. Der X-Ursprung << sec:machine-coordinate-system,Maschinenursprung>> sollte sein, wenn sich das Referenzwerkzeug (das mit dem Nullversatz) im Drehzentrum befindet. Weitere Informationen finden Sie im Abschnitt G96.

Vorschub pro Umdrehung

Vorschub pro Umdrehung (engl. feed per revolution, oder kurz FPR) bewegt die Z-Achse um den Betrag F pro Umdrehung. Dies ist nicht für das Gewindeschneiden, verwenden Sie G76 für das Gewindeschneiden. Weitere Informationen finden Sie im Abschnitt G95.

2.6.6. Bögen

Die Berechnung von Kreisbögen kann schon schwierig genug sein, ohne den Radius- und Durchmessermodus auf Drehmaschinen sowie die Ausrichtung des Maschinenkoordinatensystems zu berücksichtigen. Das Folgende gilt für Bögen im Mittelformat. Auf einer Drehmaschine sollten Sie G18 in Ihre Präambel aufnehmen, da die Voreinstellung G17 ist, auch wenn Sie sich im Drehmaschinenmodus befinden, in der Benutzeroberfläche AXIS. Bögen in der G18 XZ-Ebene verwenden I- (X-Achse) und K- (Z-Achse) Offsets.

Bögen und Drehmaschinendesign

Bei einer typischen Drehmaschine befindet sich die Spindel auf der linken Seite des Bedieners und die Werkzeuge auf der Bedienerseite der Spindelmittellinie. Die imaginäre Y-Achse (+) zeigt dabei in der Regel auf den Boden.

Für diese Art von Einrichtung gilt Folgendes:

- Die Z-Achse (+) zeigt nach rechts, weg von der Spindel.
- Die X-Achse (+) zeigt in Richtung des Bedieners, und wenn sie sich auf der Bedienerseite der Spindel befindet, sind die X-Werte positiv.

Bei einigen Drehbänken mit Werkzeugen auf der Rückseite zeigt die imaginäre Y-Achse (+) nach oben.

G2/G3 Die Richtungen der Bogen basieren auf der Achse, um die sie sich drehen. Bei Drehbänken ist das die imaginäre Y-Achse. Wenn die Y-Achse (+) auf den Boden zeigt, müssen Sie nach oben schauen, damit der Bogen in die richtige Richtung zu gehen scheint. Wenn Sie also von oben schauen, kehren Sie die G2/G3 um, damit der Bogen in die richtige Richtung zu gehen scheint.

Radius & Durchmesser-Modus

Bei der Berechnung von Bögen im Radiusmodus müssen Sie sich nur die Drehrichtung merken, die für Ihre Drehmaschine gilt.

Bei der Berechnung von Bögen im Durchmessermodus ist X der Durchmesser und der X-Offset (I) der Radius, selbst wenn Sie sich im G7-Durchmessermodus befinden.

2.6.7. Werkzeugpfad

Kontrollpunkt

Der Kontrollpunkt für das Werkzeug folgt der programmierten Bahn. Der Kontrollpunkt ist der Schnittpunkt einer Linie, die parallel zur X- und Z-Achse verläuft und den Durchmesser der Werkzeugspitze tangiert, wie er beim Antasten der X- und Z-Achse für dieses Werkzeug definiert wurde. Beim Drehen oder Plandrehen von Teilen mit geraden Seiten folgen die Schneidbahn und die Werkzeugschneide demselben Weg. Beim Drehen von Radien und Winkeln folgt die Kante der Werkzeugspitze nicht der programmierten Bahn, es sei denn, die Fräskompensation ist aktiviert. In den folgenden Abbildungen können Sie sehen, dass der Kontrollpunkt nicht der Werkzeugkante folgt, wie Sie vielleicht annehmen.

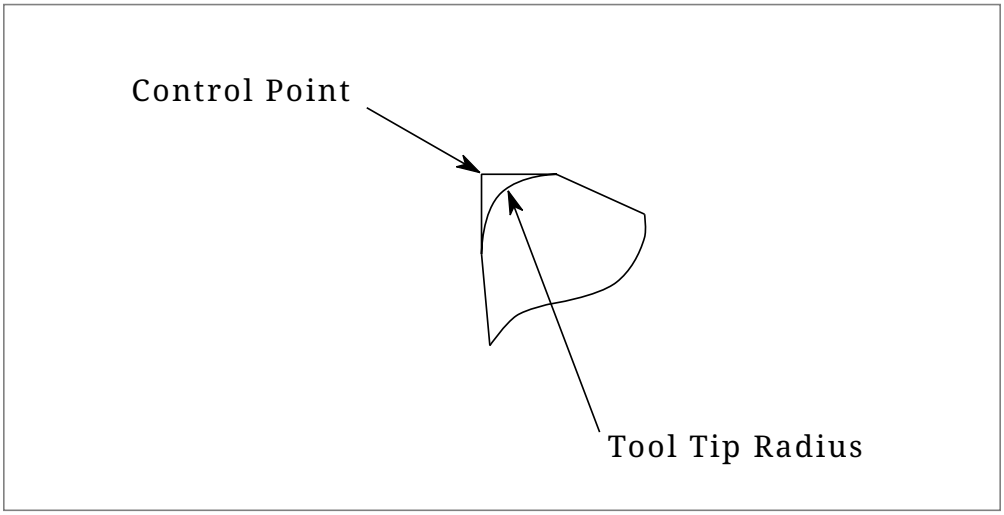


Figure 20. Kontrollpunkt

Schneidwinkel ohne Fräser Compensation

Stellen Sie sich nun vor, wir programmieren eine Rampe ohne Schneidwerkzeugs-Kompensation (engl. kurz cutter comp). Die programmierte Bahn ist in der folgenden Abbildung dargestellt. Wie Sie in der Abbildung sehen können, sind die programmierte Bahn und die gewünschte Schnittbahn ein und dasselbe, solange wir uns nur in X- oder Z-Richtung bewegen.

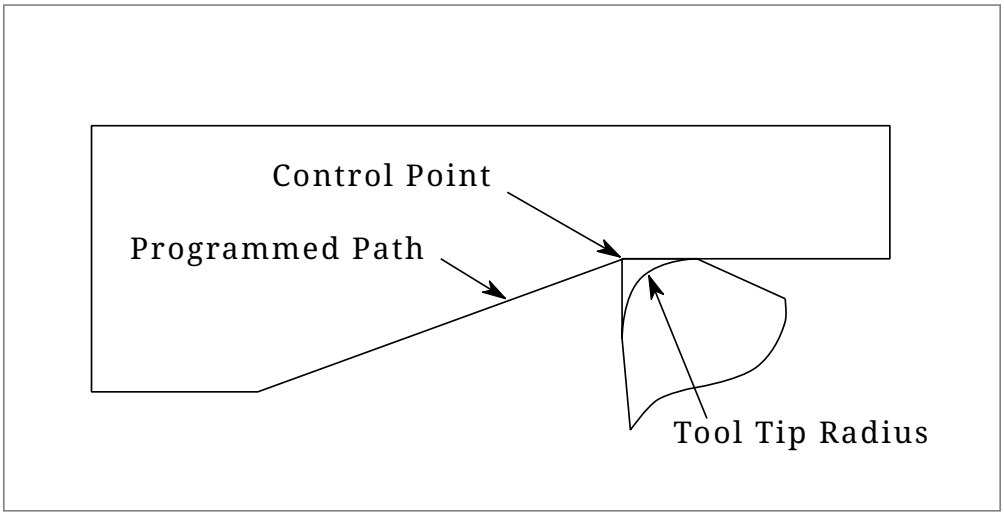


Figure 21. Rampe Eingang

Wenn sich nun der Kontrollpunkt entlang der programmierten Bahn bewegt, folgt die tatsächliche Fräskante nicht der programmierten Bahn, wie in der folgenden Abbildung dargestellt. Es gibt zwei Möglichkeiten, dieses Problem zu lösen: die Kompensation der Schneide und die Anpassung der programmierten Bahn, um den Spitzenradius zu kompensieren.

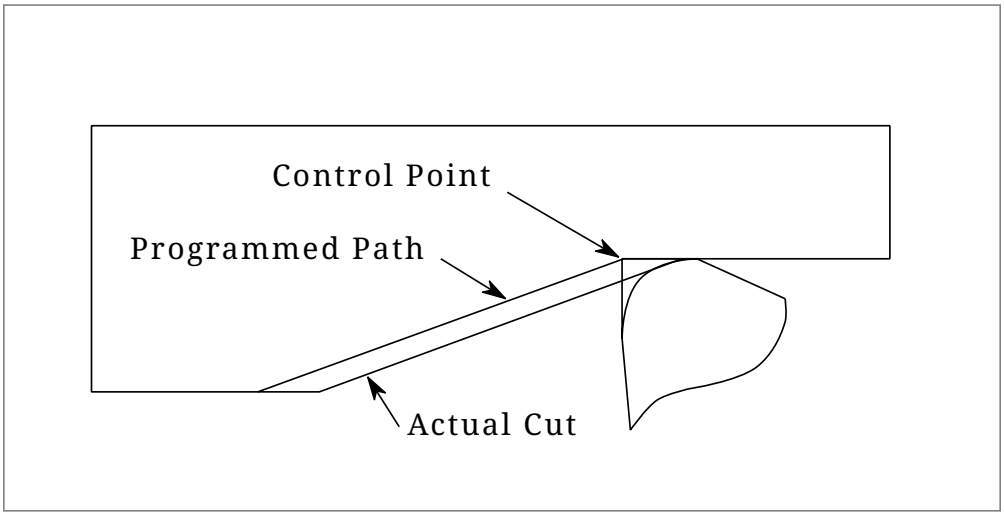


Figure 22. Rampenpfad

Im obigen Beispiel ist es eine einfache Übung, die programmierte Bahn so anzupassen, dass sie die gewünschte tatsächliche Bahn ergibt, indem die programmierte Bahn für die Rampe um den Radius der Werkzeugspitze nach links verschoben wird.

Schneiden eines Radius

In diesem Beispiel werden wir untersuchen, was bei einem Radiusschnitt ohne Fräserabgleich passiert. In der nächsten Abbildung sehen Sie, wie das Werkzeug den Außendurchmesser des Werkstücks dreht. Der Kontrollpunkt des Werkzeugs folgt der programmierten Bahn und das Werkzeug berührt den Außendurchmesser des Werkstücks.

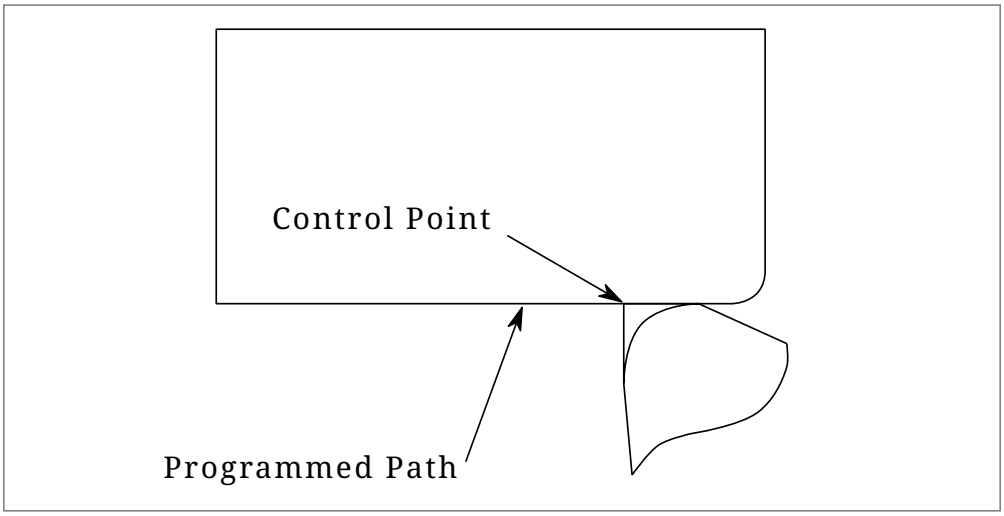


Figure 23. Drehbearbeitung (engl. turning cut)

In der nächsten Abbildung sehen Sie, wie sich das Werkzeug dem Ende des Teils nähert, der Kontrollpunkt folgt immer noch der Bahn, aber die Werkzeugspitze hat das Teil verlassen und schneidet Luft. Sie können auch sehen, dass das Teil, obwohl ein Radius programmiert wurde, tatsächlich mit einer quadratischen Ecke endet.

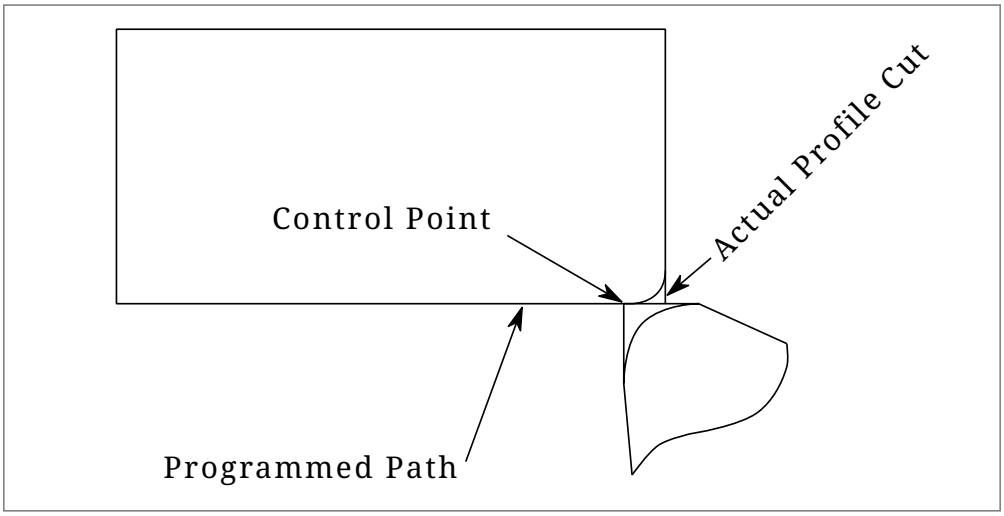


Figure 24. Radiusschnitt

Jetzt können Sie sehen, wie der Kontrollpunkt dem programmierten Radius folgt, die Werkzeugspitze hat das Teil verlassen und schneidet nun Luft.

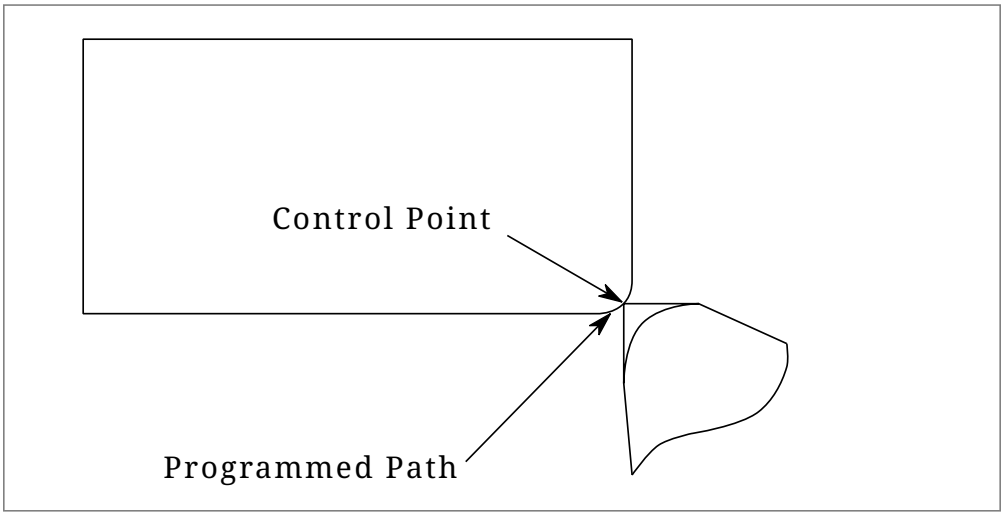


Figure 25. Radiusschnitt

In der letzten Abbildung sehen Sie, dass die Werkzeugspitze den Schnitt an der Fläche beendet, aber eine quadratische Ecke anstelle eines schönen Radius hinterlässt. Beachten Sie auch, dass, wenn Sie den Schnitt so programmieren, dass er in der Mitte des Teils endet, eine kleine Menge Material vom Radius des Werkzeugs übrig bleibt. Um einen Flächenschnitt in der Mitte des Werkstücks zu beenden, müssen Sie das Werkzeug so programmieren, dass es über die Mitte hinausgeht und mindestens den Radius der Werkzeugspitze hat.

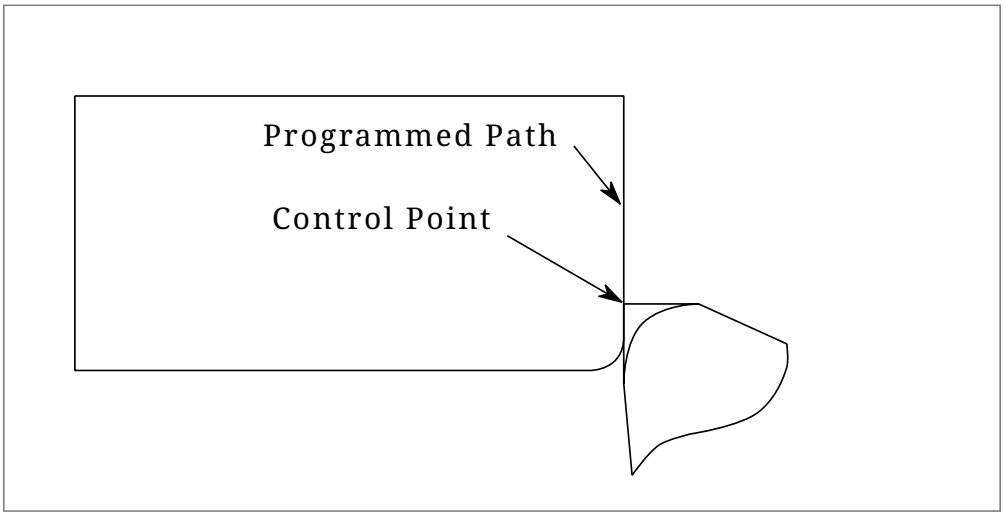


Figure 26. Face Cut

Verwenden der Fräser (engl. cutter)-Kompensation

- Bei der Verwendung von Cutter Comp auf einer Drehmaschine stellen Sie sich den Radius der Werkzeugspitze als den Radius eines runden Fräsers vor.
- Bei der Verwendung von Cutter Comp muss der Weg für ein rundes Werkzeug groß genug sein, damit es sich nicht in die nächste Linie fräst.
- Wenn Sie auf der Drehmaschine gerade Linien schneiden, möchten Sie vielleicht nicht den Cutter Comp verwenden. Wenn Sie zum Beispiel ein Loch mit einer eng anliegenden Bohrstange bohren, haben Sie möglicherweise nicht genug Platz, um die Ausfahrbewegung durchzuführen.
- Die Eintrittsbewegung in einen Cutter-Comp-Bogen ist wichtig, um die richtigen Ergebnisse zu

erzielen.

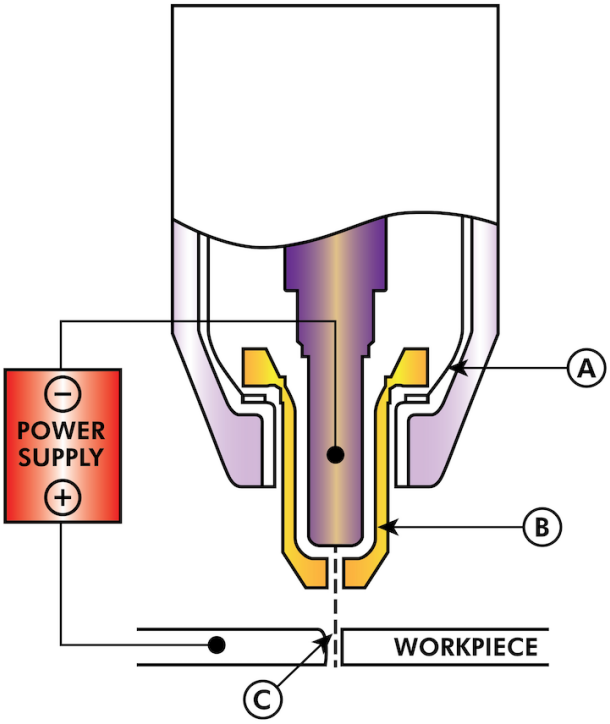
2.7. Einführung zum Plasmaschneiden LinuxCNC Benutzer

2.7.1. Was ist Plasma?

Plasma ist ein vierter Aggregatzustand, ein ionisiertes Gas, das auf eine extrem hohe Temperatur erhitzt und ionisiert wurde, so dass es elektrisch leitfähig wird. Beim Plasmaschneiden und -fugenhobeln wird dieses Plasma verwendet, um einen elektrischen Lichtbogen auf das Werkstück zu übertragen. Das zu schneidende oder zu entfernende Metall wird durch die Hitze des Lichtbogens geschmolzen und dann weggeblasen. Während das Ziel des Plasmaschneidens das Trennen des Materials ist, wird das Fugenhobeln dazu verwendet, Metalle in einer kontrollierten Tiefe und Breite zu entfernen.

Plasmabrenner sind ähnlich aufgebaut wie die Zündkerze eines Autos. Sie bestehen aus einem negativen und einem positiven Teil, die durch einen mittleren Isolator getrennt sind. Im Inneren des Brenners wird der Pilotlichtbogen in der Lücke zwischen der negativ geladenen Elektrode und der positiv geladenen Spitze gezündet. Sobald der Pilotlichtbogen das Plasmagas ionisiert hat, strömt die überhitzte Gassäule durch die kleine Öffnung in der Brennerspitze, die auf das zu schneidende Metall gerichtet ist.

In einem Plasmaschneidbrenner tritt ein kühles Gas in Zone B ein, wo ein Pilotlichtbogen zwischen der Elektrode und der Brennerspitze das Gas erhitzt und ionisiert. Der Hauptschneidlichtbogen wird dann durch die Plasmagassäule in Zone C auf das Werkstück übertragen. Indem das Plasmagas und der Lichtbogen durch eine kleine Öffnung gepresst werden, liefert der Brenner eine hohe Wärmekonzentration auf einer kleinen Fläche. Der steife, eingeschnürte Plasmalichtbogen ist in Zone C zu sehen. Beim Plasmaschneiden wird Gleichstrom (DC) mit gerader Polarität verwendet, wie in der Abbildung dargestellt. Zone A leitet ein Sekundärgas, das den Brenner kühlt. Dieses Gas unterstützt auch das Hochgeschwindigkeitsplasmagas beim Herausblasen des geschmolzenen Metalls aus dem Schnitt, wodurch ein schneller, schlackenfreier Schnitt ermöglicht wird.



TYPICAL TORCH HEAD DETAIL

2.7.2. Bogen-Initialisierung

Es gibt zwei Hauptmethoden für die Lichtbogeninitialisierung bei Plasmaschneidanlagen, die für den CNC-Betrieb ausgelegt sind. Während andere Methoden auf einigen Maschinen verwendet werden (z. B. Scratch-Start, wo physischer Kontakt mit dem Material erforderlich ist), sind sie für CNC-Anwendungen ungeeignet...

Hochfrequenzstart

Dieser Starttyp ist weit verbreitet und am längsten im Einsatz. Obwohl es sich um eine ältere Technologie handelt, funktioniert sie gut und startet schnell. Aufgrund der Hochfrequenz-Hochspannung, die zur Ionisierung der Luft erforderlich ist, hat sie jedoch einige Nachteile. Sie stört oft die umliegenden elektronischen Schaltkreise und kann sogar Bauteile beschädigen. Außerdem ist ein spezieller Schaltkreis erforderlich, um einen Pilotbogen zu erzeugen. Preiswerte Modelle verfügen nicht über einen Pilotlichtbogen und erfordern eine Berührung des Verbrauchsmaterials mit dem Werkstück, um es zu starten. Die Verwendung eines HF-Schaltkreises kann auch den Wartungsaufwand erhöhen, da es in der Regel einstellbare Punkte gibt, die von Zeit zu Zeit gereinigt und neu eingestellt werden müssen.

Blowback Start

Bei diesem Starttyp wird ein kleiner Kolben oder eine Kartusche im Brennerkopf durch den Luftdruck, der dem Schneidgerät zugeführt wird, zurückgedrückt, um einen kleinen Funken zwischen der Innenfläche des Verschleißteils zu erzeugen, der die Luft ionisiert und eine kleine Plasmaflamme erzeugt. Dadurch wird auch ein "Pilotlichtbogen" erzeugt, der für eine Plasmaflamme sorgt, die brennt,

egal ob sie mit dem Metall in Kontakt ist oder nicht. Dies ist ein sehr guter Starttyp, der inzwischen von mehreren Herstellern verwendet wird. Ihr Vorteil ist, dass sie etwas weniger Schaltkreise benötigt, recht zuverlässig ist und weitaus weniger elektrisches Rauschen erzeugt.

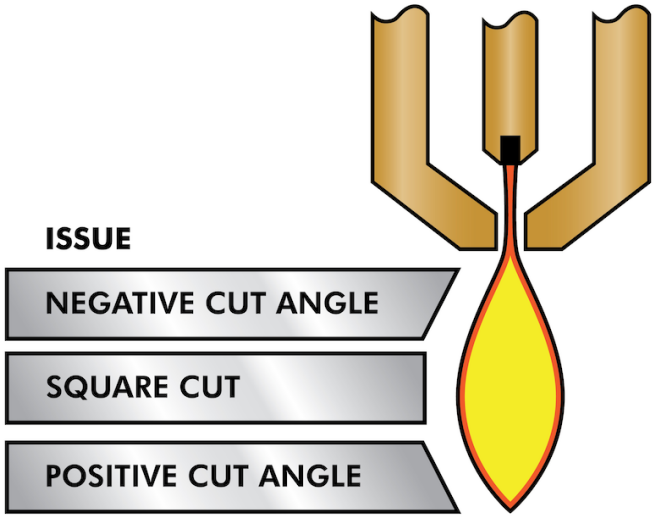
Bei CNC-Luftplasmasystemen der Einstiegsklasse wird der Blowback-Stil bevorzugt, um elektrische Interferenzen mit der Elektronik und Standard-PCs zu minimieren, aber bei größeren Maschinen ab 200 Ampère ist der Hochfrequenzstart immer noch die Regel. Diese erfordern PCs und Elektronik auf Industrieniveau, und selbst kommerzielle Hersteller hatten Probleme mit Fehlern, weil sie es versäumt haben, das elektrische Rauschen in ihren Konstruktionen zu berücksichtigen.

2.7.3. CNC-Plasma

Die Plasmabearbeitung auf CNC-Maschinen ist im Vergleich zum Fräsen oder Drehen ziemlich einzigartig und ein etwas verwaistes Verfahren. Eine ungleichmäßige Erwärmung des Materials durch den Plasmalichtbogen führt dazu, dass sich das Blech biegt und verzieht. Die meisten Bleche kommen nicht in einem sehr gleichmäßigen oder flachen Zustand aus dem Walzwerk oder der Presse. Dicke Bleche (30 mm und mehr) können um 50 mm bis 100 mm aus der Ebene geraten. Die meisten anderen CNC-G-Code-Operationen beginnen mit einer bekannten Referenz oder einem Stück Material, das eine bekannte Größe und Form hat, und der G-Code wird geschrieben, um den Überschuss abzuschruppen und dann das fertige Teil zu schneiden. Bei Plasma ist es aufgrund des unbekannten Zustands des Blechs unmöglich, einen G-Code zu erstellen, der diese Abweichungen im Material berücksichtigt.

Ein Plasmalichtbogen hat eine ovale Form, und die Schnitthöhe muss kontrolliert werden, um abgeschrägte Kanten zu minimieren. Wenn der Brenner zu hoch oder zu niedrig ist, können die Kanten übermäßig abgeschrägt werden. Es ist auch wichtig, dass der Brenner senkrecht zur Oberfläche gehalten wird.

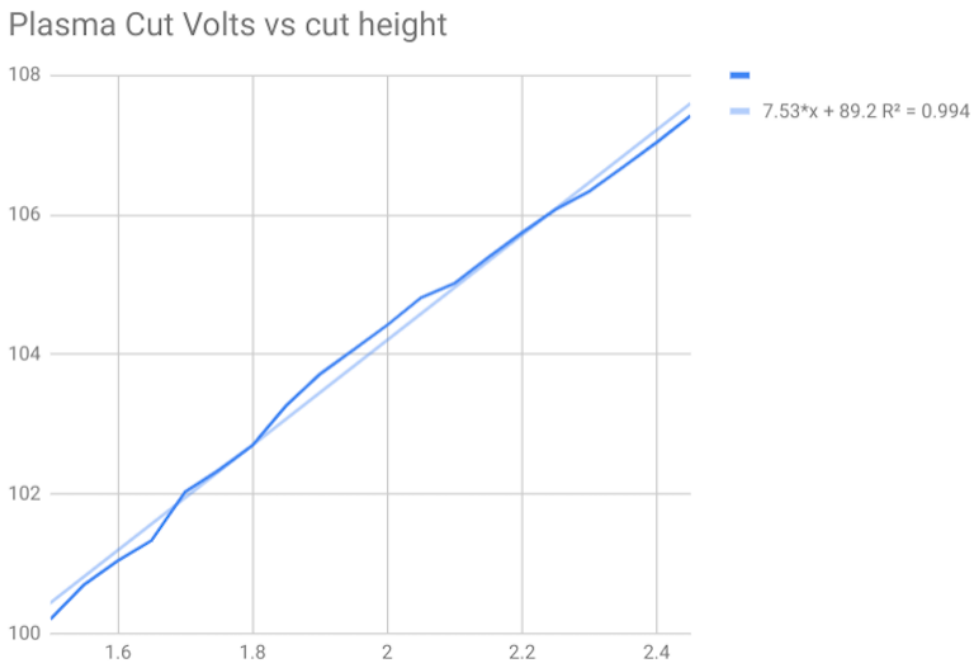
- **Abstand zwischen Brenner und Werkstück kann die Kantenfase beeinflussen**



- **Negativer Schnittwinkel:** Brenner zu niedrig, Abstand zwischen Brenner und Werkstück vergrößern.
- **Positiver Schnittwinkel:** Brenner zu hoch, Abstand zwischen Brenner und Werkstück verringern.

NOTE Eine leichte Abweichung der Schnittwinkel kann normal sein, solange sie innerhalb der Toleranz liegt.

Die Fähigkeit, die Schnitthöhe in einer solch feindlichen und sich ständig verändernden Umgebung präzise zu steuern, ist eine sehr schwierige Herausforderung. Glücklicherweise gibt es eine sehr lineare Beziehung zwischen der Brennerhöhe (Lichtbogenlänge) und der Lichtbogenspannung, wie diese Grafik zeigt.



Dieses Diagramm wurde aus einer Stichprobe von etwa 16.000 Messwerten bei unterschiedlichen Schnitthöhen erstellt, und die Regressionsanalyse zeigt 7,53 V/mm mit einer Zuverlässigkeit von 99,4 %. In diesem speziellen Fall wurde die Stichprobe von einer Everlast-Maschine mit 50 A entnommen, die von LinuxCNC gesteuert wird.

Die Brennerspannung ist dann eine ideale Prozesssteuerungsvariable für die Einstellung der Schnitthöhe. Nehmen wir der Einfachheit halber an, dass sich die Spannung um 10 V/mm ändert. Dies kann umgerechnet werden als 1 Volt pro 0,1 mm (0,004"). Die großen Plasmamaschinenhersteller (z. B. Hypertherm, Thermal Dynamics und ESAB) geben Schneidtabellen heraus, in denen die empfohlene Schneidhöhe und die geschätzte Lichtbogenspannung bei dieser Höhe sowie einige zusätzliche Daten angegeben sind. Wenn also die Lichtbogenspannung 1 V höher ist als die Herstellerangaben, muss die Steuerung den Brenner lediglich um 0,1 mm (0,004") absenken, um die gewünschte Schneidhöhe zu erreichen. Zur Steuerung dieses Prozesses wird üblicherweise eine Brennerhöhensteuerung (THC) verwendet.

2.7.4. Auswahl einer Plasmamaschine für CNC-Bearbeitungen

Auf dem Markt gibt es heute eine Vielzahl von Plasmamaschinen, von denen nicht alle für den CNC-Einsatz geeignet sind. CNC-Plasmaschneiden ist ein komplexer Vorgang, und es wird empfohlen, dass Integratoren eine geeignete Plasmamaschine auswählen. Andernfalls kann es zu stundenlanger, erfolgloser Fehlersuche kommen, wenn man versucht, das Fehlen von Funktionen zu umgehen, die viele als obligatorisch ansehen würden.

Obwohl Regeln dazu da sind, gebrochen zu werden, wenn man die Gründe für die Anwendung der Regel versteht, sind wir der Meinung, dass ein neuer Hersteller von Plasmatischen eine Maschine mit den folgenden Merkmalen auswählen sollte:

- Blowback-Start zur Minimierung der elektrischen Geräusche und zur Vereinfachung der Konstruktion
- Ein Maschinenbrenner wird bevorzugt, aber viele haben auch Handbrenner verwendet.
- Eine vollständig abgeschirmte Brennerspitze, die eine ohmsche Abtastung ermöglicht

Wenn Sie über das nötige Budget verfügen, können Sie sich für ein höherwertiges Gerät entscheiden:

- Vom Hersteller bereitgestellte Schneidtabellen, die viele Stunden und Materialabfälle bei der Kalibrierung der Schneidparameter sparen
- Potentialfreie Kontakte für ArcOK
- Klemmen für Lichtbogen-Ein (engl. Arc On)-Schalter
- Ausgang für rohe Lichtbogenspannung (engl. raw arc voltage) oder geteilte Lichtbogenspannung
- Optional eine RS485-Schnittstelle, wenn Sie einen Hypertherm-Plasma-Cutter verwenden und diesen von der LinuxCNC-Konsole aus steuern möchten.
- Höhere Einschaltzyklen

In jüngster Zeit ist eine andere Maschinenklasse, die einige dieser Funktionen enthält, für rund 550 US-Dollar erhältlich geworden. Ein Beispiel ist der Herocut55i, der bei Amazon erhältlich ist, aber es gibt noch kein Feedback von Benutzern. Diese Maschine verfügt über einen Blasbrenner, ArcOK-Ausgang, Brennerstartkontakte und rohe Lichtbogenspannung.

2.7.5. Arten der Brennerhöhensteuerung

Die meisten THC-Einheiten sind externe Geräte und viele haben eine ziemlich grobe "Bit Bang" -Anpassungsmethode. Sie liefern zwei Signale zurück an die LinuxCNC-Steuerung. Einer schaltet sich ein, wenn sich die Z-Achse nach oben bewegen soll, und der andere schaltet sich ein, wenn sich die Z-Achse nach unten bewegen soll. Keines der beiden Signale ist wahr, wenn sich die Taschenlampe auf der richtigen Höhe befindet. Das beliebte Proma 150 THC ist ein Beispiel für diese Art von THC. Die LinuxCNC THCUD-Komponente wurde entwickelt, um mit dieser Art von THC zu arbeiten.

Mit der Veröffentlichung der Mesa THCAD Spannung zu Frequenz-Schnittstelle, war LinuxCNC in der Lage, die tatsächliche Brennerspannung über einen Encoder-Eingang zu decodieren. Dies ermöglichte LinuxCNC, um die Z-Achse zu steuern und zu beseitigen externe Hardware. Frühe Implementierungen unter Verwendung der THCAD repliziert die "Bit-Bang"-Ansatz. Die LinuxCNC THC Komponente ist ein Beispiel für diesen Ansatz.

Jim Colt von Hypertherm hat zu Protokoll gegeben, dass die besten THC-Controller vollständig in die CNC-Steuerung selbst integriert wurden. Natürlich bezog er sich auf High-End-Systeme, die von Hypertherm, Esab, Thermal Dynamics und anderen wie Advanced Robotic Technology in Australien hergestellt wurden, und träumte nicht davon, dass Open Source mit diesem Ansatz Systeme produzieren könnte, die mit High-End-Systemen konkurrieren.

Die Einbeziehung externer Offsets in Linuxcnc V2.8 ermöglichte es der Plasmasteuerung in LinuxCNC, auf ein ganz neues Niveau zu steigen. Externe Offsets beziehen sich auf die Möglichkeit, einen Versatz auf die Achsenbefehlsposition außerhalb des Motion Controllers anzuwenden. Dies ist perfekt für die Plasma-THC-Kontrolle als Methode zur Anpassung der Brennerhöhe in Echtzeit basierend auf unserer gewählten Prozesssteuerungsmethodik. Nach einer Reihe von experimentellen Builds wurde die Plasmac-Konfiguration in LinuxCNC 2.8 integriert. QtPlasmaC hat Plasmac in LinuxCNC 2.9 abgelöst. Dies war ein äußerst ehrgeiziges Projekt und viele Menschen auf der ganzen Welt waren am Testen und Verbessern des Funktionsumfangs beteiligt. QtPlasmaC ist insofern einzigartig, als sein Designziel darin bestand, alle THCs zu unterstützen, einschließlich der einfachen Bit-Bang-THCs bis hin zur ausgeklügelten Brennerspannungssteuerung, wenn die Spannung LinuxCNC über einen THCAD oder einen anderen Spannungssensor zur Verfügung gestellt wird. Darüber hinaus ist QtPlasmaC als eigenständiges System konzipiert, das keine zusätzlichen G-Code-Subroutinen benötigt und es dem Benutzer ermöglicht, eigene Schnittdiagramme zu definieren, die im System gespeichert sind und über ein Dropdown-Menü zugänglich sind.

2.7.6. Lichtbogen-OK-Signal

Plasmageräte mit einer CNC-Schnittstelle enthalten eine Reihe von Trockenkontakten (z. B. ein Relais), die sich schließen, wenn ein verlässlicher Lichtbogen entsteht, und jede Seite dieser Kontakte ist mit Stiften an der CNC-Schnittstelle verbunden. Der Erbauer eines Plasmatisches sollte eine Seite dieser Stifte mit der Feldspannung und die andere mit einem Eingangsstift verbinden. Auf diese Weise kann die CNC-Steuerung erkennen, wann ein gültiger Lichtbogen entstanden ist und wann er unerwartet verloren geht. Hier gibt es eine potenzielle Falle, wenn es sich bei dem Eingang um eine Schaltung mit hoher Impedanz handelt, wie z. B. eine Mesa-Karte. Handelt es sich bei den potentialfreien Kontakten um ein einfaches Relais, ist die Wahrscheinlichkeit groß, dass der durch das Relais fließende Strom unter der Mindeststromspezifikation liegt. Unter diesen Bedingungen kann es bei den Relaiskontakten zu einer Oxidbildung kommen, die mit der Zeit zu einem intermittierenden Betrieb der Kontakte führen kann. Um dies zu verhindern, sollte ein Pull-Down-Widerstand am Eingangsstift des Reglers installiert werden. Es sollte darauf geachtet werden, dass dieser Widerstand so gewählt wird, dass der Mindeststrom durch das Relais fließt und er eine ausreichende Wattzahl hat, um die Leistung im Schaltkreis zu bewältigen. Schließlich sollte der Widerstand so montiert werden, dass die entstehende Wärme während des Betriebs keine Schäden verursacht.

Wenn Sie ein ArcOK-Signal haben, wird empfohlen, es über jedes synthetisierte Signal hinaus zu verwenden, um potenzielle Build-Probleme zu beseitigen. Ein synthetisiertes Signal, das von einem externen THC- oder QtPlasmaC-Modus 0 verfügbar ist, kann die ArcOK-Schaltung in einem Plasmawechselrichter nicht vollständig ersetzen. Einige Build-Probleme wurden beobachtet, bei denen eine Fehlkonfiguration oder Inkompatibilität mit dem Plasma-Wechselrichter durch ein synthetisiertes ArcOK-Signal aufgetreten ist. Im Großen und Ganzen ist jedoch ein korrekt konfiguriertes synthetisiertes ArcOK-Signal in Ordnung.

Ein einfaches und effektives ArcOK-Signal kann mit einem einfachen Reedrelais erreicht werden. Wickeln Sie 3 Umdrehungen eines der dicken Kabel des Plasma-Cutters drarum, z. B. das Materialklemmkabel. Setzen Sie das Relais zum Schutz in ein altes Stiftrohr ein und schließen Sie eine Seite des Relais an die Feldstromversorgung und das andere Ende an den ArcOK-Eingangspin an.

2.7.7. Erfassung der Anfangshöhe

Da die Schnitthöhe ein so kritischer Systemparameter ist und die Materialoberfläche von Natur aus uneben ist, benötigt ein Z-Achsen-Mechanismus eine Methode, um die Materialoberfläche zu erfassen. Es gibt drei Methoden, um dies zu erreichen:

1. Stromabtastung zur Erkennung eines erhöhten Motordrehmoments,
2. ein "Schwimmer"-Schalter und ein elektrischer oder
3. Ein "ohmscher" Messkreis, der geschlossen wird, wenn der Brennerschild das Material berührt.

Strommessung ist keine praktikable Technik für DIY-Tabellen, aber Schwimmerschalter und ohmsche Messung werden weiter unten besprochen:

Gleitende Schalter (engl. float switches)

Der Brenner ist auf einem Gleittisch montiert, der sich nach oben bewegen kann, wenn die Brennerspitze die Materialoberfläche berührt und einen Schalter oder Sensor auslöst. Oft wird dies unter G-Code-Steuerung mit den G38-Befehlen erreicht. Wenn dies der Fall ist, wird empfohlen, nach der ersten Sondierung von der Oberfläche wegzutasten, bis das Sondensignal mit einer langsameren Geschwindigkeit verloren geht. Stellen Sie außerdem sicher, dass die Schalterhysterese berücksichtigt wird.

Unabhängig von der verwendeten Sondierungsmethode wird dringend empfohlen, einen gleitenden Schalter einzubauen, damit ein Ausweich- oder Sekundärsignal vorhanden ist, um eine Beschädigung des Brenners bei einem Absturz zu vermeiden.

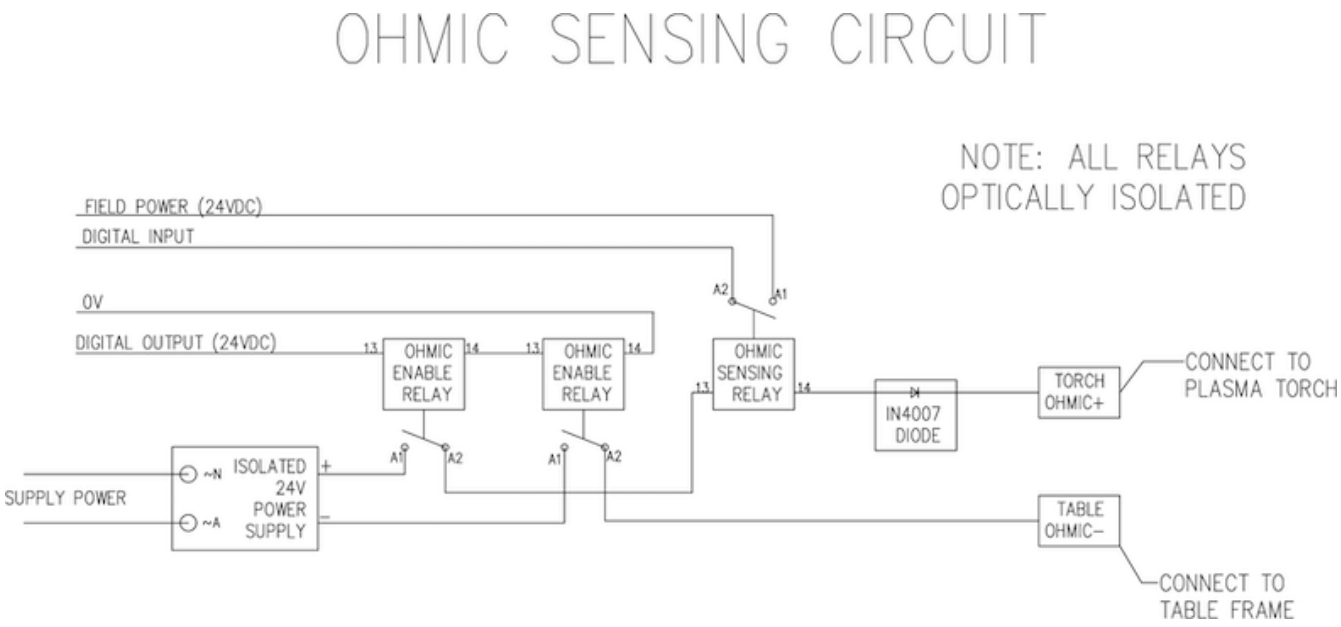
Ohmsche Erfassung

Die ohmsche Abtastung beruht auf dem Kontakt zwischen dem Brenner und dem Material, der wie ein Schalter wirkt, um ein elektrisches Signal zu aktivieren, das von der CNC-Steuerung erfasst wird. Unter der Voraussetzung, dass das Material sauber ist, kann dies eine viel genauere Methode zur Abtastung des Materials sein als ein Schwimmerschalter, der eine Ablenkung der Materialoberfläche verursachen kann. Dieser ohmsche Abtastkreis arbeitet in einer extrem feindlichen Umgebung, so dass eine Reihe von Ausfallsicherungen implementiert werden müssen, um die Sicherheit sowohl der CNC-Elektronik als auch des Bedieners zu gewährleisten. Beim Plasmaschneiden ist die am Material befestigte Erdungsklemme positiv und der Brenner ist negativ. Es wird empfohlen, dass:

1. Die ohmsche Abtastung kann nur dann eingesetzt werden, wenn der Brenner eine Abschirmung hat, die von der Brennerspitze isoliert ist, die den Schneidlichtbogen leitet.
2. Der ohmsche Schaltkreis verwendet eine völlig getrennte, isolierte Stromversorgung, die ein optoisoliertes Relais aktiviert, damit das Abtastsignal an die CNC-Steuerung übertragen werden kann.
3. Die positive Seite des Stromkreises sollte am Brenner liegen.
4. Beide Seiten des Stromkreises müssen durch optoisolierte Relais isoliert werden, bis die Messung durchgeführt wird.
5. Es müssen Sperrdioden verwendet werden, um zu verhindern, dass Lichtbogenspannung in den

ohmschen Messkreis gelangt.

Im Folgenden finden Sie eine Beispielschaltung, die sich bewährt hat und mit der LinuxCNC QtPlasmaC-Konfiguration kompatibel ist.



Hypersensing mit einem MESA THCAD-5

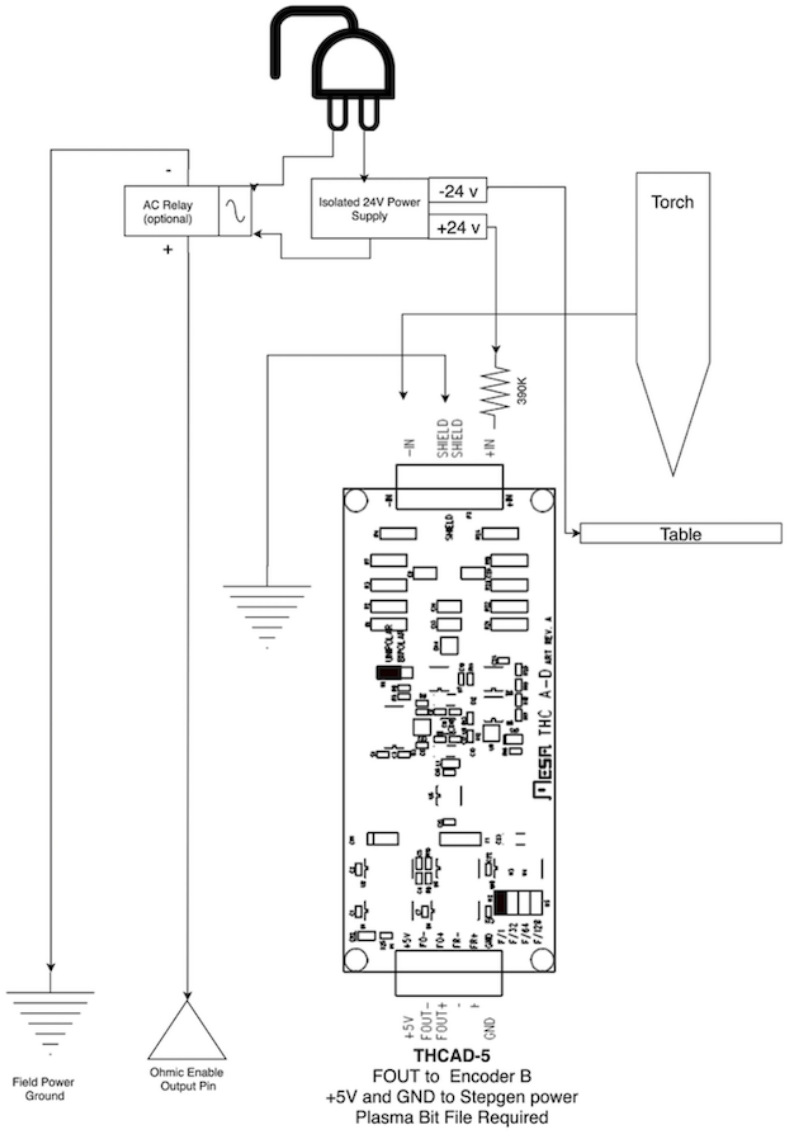
Eine ausgefeiltere Methode der Materialerkennung, bei der die Relais und Dioden entfallen, ist die Verwendung eines weiteren THCAD-5 zur Überwachung der Spannung des Materialerkennungsschaltkreises über eine isolierte Stromversorgung. Dies hat den Vorteil, dass das THCAD für die feindliche elektrische Umgebung des Plasmas ausgelegt ist und die Logikseite vollständig und sicher von der Hochspannungsseite isoliert.

Um diese Methode zu implementieren, ist ein zweiter Encodereingang erforderlich.

Bei Verwendung einer Mesa-Karte ist eine andere Firmware verfügbar, die 2 zusätzliche Encoder A-Eingänge an den Pins Encoder B und Encoder Index bereitstellt. Diese Firmware kann für die Karten 7I76E und 7I96 von der Mesa-Website auf den Produktseiten heruntergeladen werden.

Der THCAD ist empfindlich genug, um den Anstieg der Schaltungsspannung bei steigendem Kontaktdruck zu erkennen. Die in LinuxCNC enthaltene Komponente ohmic.comp kann die Abtastspannung überwachen und einen Spannungsschwellenwert festlegen, bei dessen Überschreitung davon ausgegangen wird, dass ein Kontakt hergestellt und ein Ausgang aktiviert ist. Durch die Überwachung der Spannung kann ein niedrigerer Schwellenwert für die Unterbrechung des Stromkreises festgelegt werden, um eine starke Schalterhysterese einzubauen. Dadurch werden Fehlauslösungen minimiert. In unseren Tests haben wir festgestellt, dass die Materialerfassung mit dieser Methode empfindlicher und robuster ist und die Verdrahtung einfacher zu realisieren ist. Ein weiterer Vorteil der Verwendung von Softwareausgängen anstelle von physischen E/A-Pins ist, dass dadurch Pins für andere Zwecke frei werden. Dieser Vorteil ist hilfreich, um das Beste aus dem Mesa 7I96 herauszuholen, das über begrenzte E/A-Pins verfügt.

Der folgende Schaltplan zeigt, wie eine Hypersensing-Schaltung realisiert werden kann.



Wir haben eine 15 W Mean Well HDR-15 Ultra Slim DIN Rail Supply 24 V DIN-Schienen-basierte isolierte Stromversorgung verwendet. Dies ist ein doppelt isoliertes Gerät der Isolationsklasse II, das jeder Lichtbogenspannung standhält, die an die Klemmen angelegt werden könnte.

Beispiel HAL-Code für Hypersensing

Der folgende HAL-Code kann in die custom.hal von QtPlasmaC eingefügt werden, um die ohmsche Abtastung am Encoder 2 einer 7I76E zu aktivieren. Installieren Sie die richtige Bit-Datei und schließen Sie den THCAD an IDX+ und IDX- an. Stellen Sie sicher, dass die Kalibrierungseinstellungen mit denen Ihres THCAD-5 übereinstimmen.

```
# --- Load the Component ---
loadrt ohmic names=ohmicsense
```

```
addf ohmicsense servo-thread

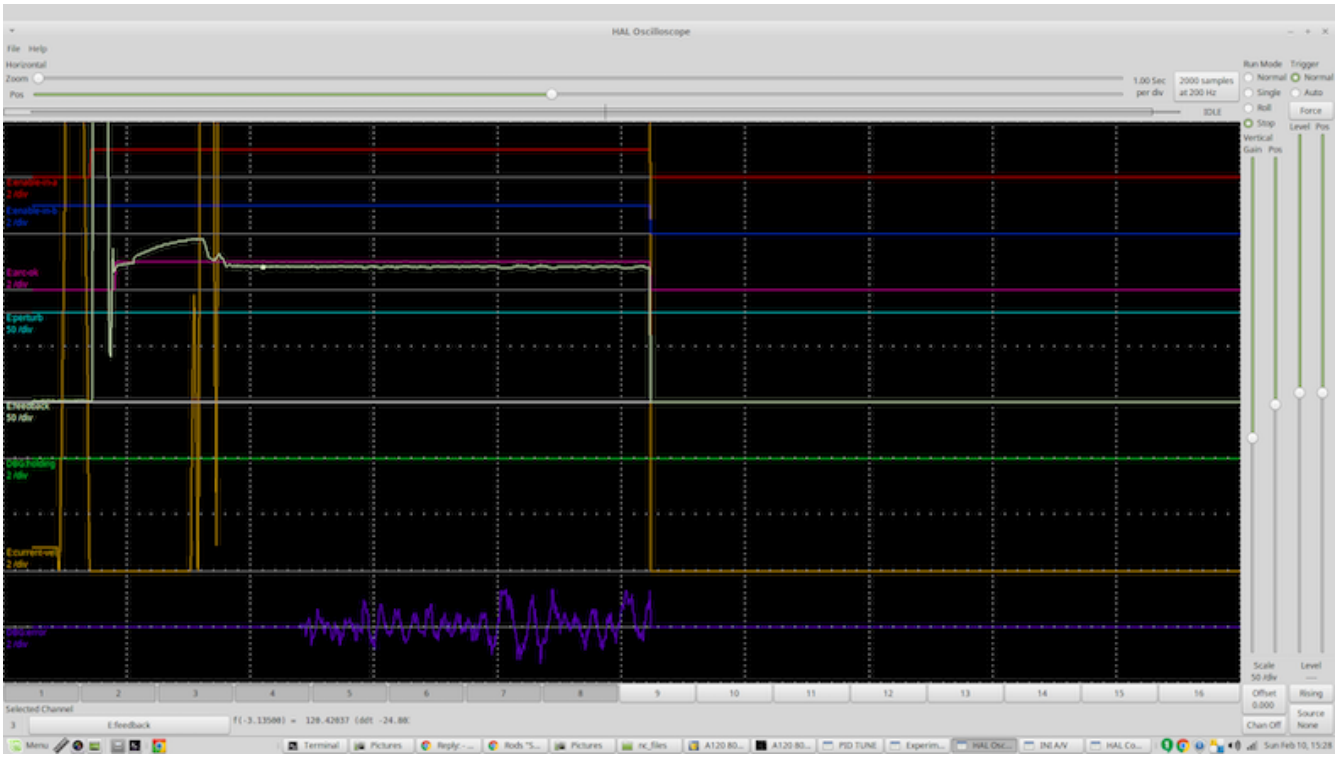
# --- 7I76E ENCODER 2 SETUP FOR OHMIC SENSING---
setp hm2_7i76e.0.encoder.02.scale -1
setp hm2_7i76e.0.encoder.02.counter-mode 1

# --- Configure the component ---
setp ohmicsense.thcad-0-volt-freq    140200
setp ohmicsense.thcad-max-volt-freq  988300
setp ohmicsense.thcad-divide         32
setp ohmicsense.thcad-fullscale      5
setp ohmicsense.volt-divider         4.9
setp ohmicsense.ohmic-threshold      22.0
setp ohmicsense.ohmic-low            1.0
net ohmic-vel ohmicsense.velocity-in <= hm2_7i76e.0.encoder.02.velocity

# --- Replace QtPlasmaC's Ohmic sensing signal ---
unlinkp db_ohmic.in
net ohmic-true ohmicsense.ohmic-on => db_ohmic.in
net plasmac:ohmic-enable    =>  ohmicsense.is-probing
```

2.7.8. THC-Verzögerung

Wenn ein Lichtbogen entsteht, steigt die Lichtbogenspannung deutlich an und pendelt sich dann wieder auf eine stabile Spannung auf Höhe des Schnittes ein. Wie die grüne Linie in der Abbildung unten zeigt.



Es ist wichtig, dass die Plasmasteuerung "abwartet", bevor sie die Brennerspannung automatisch abtastet und mit der THC-Regelung beginnt. Wenn sie zu früh aktiviert wird, liegt die Spannung über der gewünschten Abschaltspannung und der Brenner wird heruntergefahren, um einen vermeintlichen Überhöhungszustand zu beheben.

In unseren Tests variiert dies je nach Maschine und Material zwischen 0,5 und 1,5 s. Daher ist eine

Verzögerung von 1,5 Sekunden nach einem gültigen arcOK Signal empfangen wird, bevor die THC-Steuerung ist eine sichere Grundeinstellung. Wenn Sie dies für ein bestimmtes Material zu verkürzen wollen, wird LinuxCNC Halscope können Sie die Brennerspannung zu plotten und fundierte Entscheidungen über die kürzeste sichere Verzögerung verwendet wird.

NOTE Liegt die Schnittgeschwindigkeit am Ende dieser Verzögerung noch nicht in der Nähe der gewünschten Schnittgeschwindigkeit, sollte die Steuerung warten, bis diese erreicht ist, bevor sie die THC aktiviert.

2.7.9. Abtastung der Brennerspannung

Anstatt sich auf die Schneidtabellen des Herstellers zu verlassen, um die gewünschte Brennerspannung einzustellen, ziehen es viele Leute (einschließlich des Verfassers) vor, die Spannung zu messen, wenn die THC aktiviert ist, und diese als Sollwert zu verwenden.

2.7.10. Brenner Behinderung (engl. torch breakaway)

Es wird empfohlen, einen Mechanismus vorzusehen, der es dem Brenner ermöglicht, im Falle eines Aufpralls auf das Material oder eines hochgekippten Schneidteils "abzukommen" oder abzufallen. Es sollte ein Sensor installiert werden, damit die CNC-Steuerung erkennen kann, ob dies geschehen ist, und das laufende Programm anhalten kann. Normalerweise wird eine Abreißsicherung mit Magneten realisiert, um den Brenner am Z-Achsentisch zu befestigen.

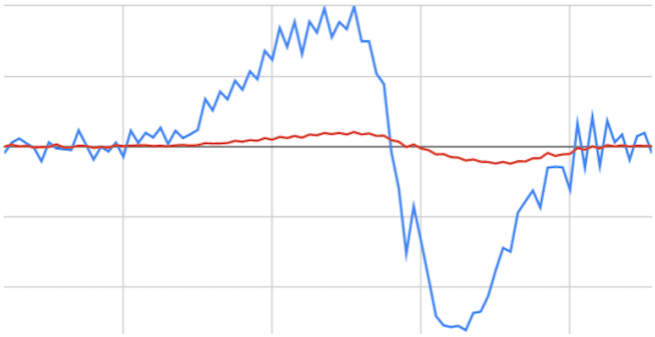
2.7.11. Eckensicherung (engl. corner lock) / Verhindern von Eintauchen bei Geschwindigkeitsänderung (engl. velocity anti-dive)

Die LinuxCNC Trajektorie Planer ist verantwortlich für die Übersetzung Geschwindigkeit und Beschleunigung Befehle in Bewegung, die den Gesetzen der Physik gehorchen. Zum Beispiel wird die Bewegung zu verlangsamen, wenn die Verhandlung einer Ecke. Während dies bei Fräsmaschinen oder Oberfräsen kein Problem ist, stellt dies beim Plasmaschneiden ein besonderes Problem dar, da die Lichtbogenspannung bei langsamerer Bewegung steigt. Dies führt dazu, dass die THC den Brenner herunterfährt. Einer der enormen Vorteile einer THC-Steuerung innerhalb der LinuxCNC Motion Controller eingebettet ist, dass es weiß, was los ist zu allen Zeiten. So wird es eine triviale Angelegenheit, um die aktuelle Geschwindigkeit (`motion.current-velocity`) zu überwachen und THC-Betrieb zu halten, wenn es unter einen bestimmten Schwellenwert fällt (zB 10% unter dem gewünschten Vorschub).

2.7.12. Hohlraum- / Schnitt-Kreuzung (engl. void/kerf crossing)

Wenn der Plasmabrenner beim Schneiden über einen Hohlraum fährt, steigt die Lichtbogenspannung schnell an und die THC reagiert mit einer heftigen Abwärtsbewegung, die den Brenner in das Material schlagen und es möglicherweise beschädigen kann. Dies ist eine Situation, die schwer zu erkennen und zu handhaben ist. Bis zu einem gewissen Grad kann dies durch gute Verschachtelungstechniken gemildert werden, aber bei dickerem Material kann es immer noch vorkommen, dass etwas Schlacke herausfällt. Dies ist das eine Problem, das noch innerhalb der LinuxCNC Open-Source-Bewegung gelöst werden muss.

Ein vorgeschlagenes Verfahren ist die Überwachung der Änderungsrate der Brennerspannung über die Zeit (dv/dt), da dieser Parameter beim Durchqueren eines Hohlraums um Größenordnungen höher ist als bei normaler Verformung des Materials. Das folgende Diagramm zeigt eine niedrig aufgelöste Darstellung von dv/dt (in blau) beim Durchqueren eines Hohlraums. Die rote Kurve ist ein gleitender Durchschnitt der Brennerspannung.



So sollte es möglich sein, den gleitenden Durchschnitt mit dem dv/dt zu vergleichen und THC-Betrieb zu stoppen, sobald die dv/dt den normalen Bereich aufgrund von Verzug erwartet überschreitet. Es muss noch mehr Arbeit in diesem Bereich investiert werden, damit es zu einer funktionierenden Lösung in LinuxCNC kommt.

2.7.13. Schneiden von Löchern und kleinen Formen

Beim Schneiden von Löchern und kleinen Formen empfiehlt es sich, den Schnitt zu verlangsamen.

John Moore sagt: "Wenn Sie Einzelheiten über das Schneiden von präzisen kleinen Löchern erfahren möchten, sehen Sie sich die Verkaufsunterlagen über die *True Hole Technology* von Hypertherm an, und schauen Sie auch auf PlasmaSpider nach, der Benutzer seanp hat ausführlich über seine Arbeit mit einfachem Luftplasma berichtet.

Die allgemein anerkannte Methode, um mit einem Luftplasma gute Löcher von 37 mm Durchmesser bis hinunter zur Materialstärke mit minimaler Verjüngung zu erhalten, ist:

1. Verwenden Sie den empfohlenen Schneidstrom für Verbrauchsmaterialien.
2. Verwenden Sie die feste (kein THC) empfohlene Schnitthöhe für Verbrauchsmaterialien.
3. Reduzierung von 60 % auf 70 % der empfohlenen Vorschubgeschwindigkeit von Verbrauchsmaterialien und Materialien.
4. Beginnen Sie die Einfahrt in oder nahe der Mitte des Lochs.
5. Senkrechte Zuführung verwenden.
6. Kein Ausbrennen, entweder ein leichtes Überbrennen (engl. over burn) oder frühes Abfackeln (engl. early torch off), je nachdem, was für Sie am besten funktioniert.

Sie müssen experimentieren, um die exakte Lochgröße zu erhalten, da die Schnittfuge (engl. kerf) bei dieser Methode breiter ist als beim üblichen geraden Schnitt."

Diese Verlangsamung kann erreicht werden, indem Sie die Vorschubrate direkt in Ihrem Postprozessor manipulieren oder indem Sie einen adaptiven Vorschub und einen analogen Pin als Eingang verwenden.

Auf diese Weise können Sie M67/M68 verwenden, um den Prozentsatz des gewünschten Vorschubs einzustellen, bei dem geschnitten werden soll.

- Den Vorschubsgeschwindigkeit kennen

Aus der vorangegangenen Diskussion geht hervor, dass der Plasmaregler die vom Benutzer eingestellte Vorschubgeschwindigkeit kennen muss. Dies stellt ein Problem mit LinuxCNC dar, da die Vorschubrate von LinuxCNC nicht gespeichert wird, nachdem der G-Code gepuffert und analysiert wurde. Es gibt zwei Ansätze, um dies zu umgehen:

1. Ordnen Sie den F-Befehl neu zu und speichern Sie die Befehlsvorschubrate im G-Code über einen M67/M68-Befehl.
2. Speichern der Schnittdiagramme in der Plasmasteuerung und Abfrage der aktuellen Vorschubrate durch das G-Code-Programm (wie QtPlasmaC).

Eine neu zu LinuxCNC 2.9 hinzugefügte Funktion, die für das Plasmaschneiden nützlich ist, sind die Status-Tags. Dadurch wird ein „Tag“ hinzugefügt, das der Bewegung zur Verfügung steht und die aktuellen Vorschübe und Geschwindigkeiten für alle aktiven Bewegungsbefehle enthält.

2.7.14. I/O-Pins für Plasma-Controller

Plasmaschneider benötigen mehrere zusätzliche Pins. In LinuxCNC gibt es keine festen Regeln darüber, welcher Pin was macht. In dieser Diskussion gehen wir davon aus, dass der Plasmawechselrichter über eine CNC-Schnittstelle verfügt und die Controllerkarte über aktive hohe Eingänge verfügt (z. B. Mesa 7I76E).

Plasmatische können große Maschinen sein und wir empfehlen Ihnen, sich die Zeit zu nehmen, separate Max/Min-Endschalter und Zielsuchschalter für jedes Gelenk zu installieren. Die Ausnahme könnte die untere Grenze der Z-Achse sein. Wenn ein Zielsuchschalter ausgelöst wird, verzögert sich das Gelenk ziemlich langsam für maximale Genauigkeit. Das heißt, wenn Sie Zielsuchgeschwindigkeiten verwenden möchten, die der Tischgröße entsprechen, können Sie den ursprünglichen Auslösepunkt um 50-100 mm überschreiten. Wenn Sie einen gemeinsamen Home-/Endschalter verwenden, müssen Sie den Sensor mit dem letzten HOME_OFFSET vom Auslösepunkt entfernen, oder Sie lösen einen Endschalterfehler aus, wenn die Maschine aus dem Homing kommt. Dies bedeutet, dass Sie 50 mm oder mehr Achsweg mit gemeinsamen Home-/Endschaltern verlieren können. Dies geschieht nicht, wenn separate Haus- und Endschalter verwendet werden.

Die folgenden Pins sind in der Regel erforderlich (beachten Sie, dass vorgeschlagene Verbindungen möglicherweise nicht für eine QtPlasmaC-Konfiguration geeignet sind):

Lichtbogen (engl. Arc) OK (input)

- Der Wechselrichter schließt potenzialfreie Kontakte, wenn ein gültiger Lichtbogen hergestellt wird
- Schließen Sie die Feldspannung an eine ArcOK-Klemme des Wechselrichters an.
- Schließen Sie andere OK-Klemmen des Wechselrichters an den Eingangspin an.
- Normalerweise verbunden mit einem der ``motion.digital-``<*nn*> Pins für die Verwendung von G-Code mit M66

Brenner an (Ausgang)

- Löst ein Relais aus, um den Brenner-Einschalter im Inverter zu schließen.
- Verbinden Sie die Brennerklemmen am Inverter mit den Relaisausgangsklemmen.
- Verbinden einer Seite der Spule mit dem Ausgangspin.
- Verbinden Sie die andere Seite der Spule mit der Masse der Feldversorgung.
- Wenn ein mechanisches Relais verwendet wird, schließen Sie eine Rücklaufdiode (z. B. IN400x-Serie) über die Spulenanschlüsse an, wobei das Band auf der Diode zum Ausgangspin zeigt.
- Bei Verwendung eines Solid State Relais muss ggf. die Polarität an den Ausgängen beachtet werden.
- Unter bestimmten Umständen kann das integrierte Spindelrelais auf einer Mesa-Karte anstelle eines externen Relais verwendet werden.
- Normalerweise mit `spindle.0.on` verbunden.

WARNING	Es wird dringend empfohlen, dass der Brenner nicht aktiviert werden kann, während dieser Pin falsch ist, da der Brenner sonst nicht gelöscht wird, wenn der Notaus (engl. estop) gedrückt wird.

Gleitender Schalter (engl. float switch) (input)

- Wird für die Oberflächensondierung verwendet. Ein Sensor oder Schalter, der aktiviert wird, wenn der Brenner nach oben gleitet, wenn er auf das Material trifft.
- Schließen Sie den Ausgang des Näherungssensors an den ausgewählten Eingangspin an. Wenn mechanische Schalter verwendet werden. Schließen Sie eine Seite des Schalters an die Feldleistung und die andere Seite des Schalters an den Eingang an.
- Normalerweise mit `motion.probe-input` verbunden.

Ohmscher Sensor aktivieren (Ausgang)

- Siehe den Schaltplan ohmic sensing.
- Verbinden Sie den Ausgangspin mit einer Seite des Trennrelais und die andere Seite mit der Masse der Feldversorgung.
- In einer Nicht-QtPlasmaC-Konfiguration in der Regel ausgelöst durch einen ``motion.digital-out-``<*nn*>, so dass er im G-Code von `M62`/`M63`/`M64`/`M65` gesteuert werden kann.

Ohmsche Sensorik (engl. ohmic sensing) (Eingang)

- Beachten Sie das zuvor gezeigte Schema zu ohmic sensing.
- Eine isolierte Stromversorgung löst ein Relais aus, wenn der Brennerschild das Material berührt.
- Schließen Sie die Feldspannung an eine Ausgangsklemme und die andere an den Eingang an.
- Achten Sie auf die Polarität der Relais, wenn optoentkoppelte Halbleiterrelais verwendet werden.
- Üblicherweise an `motion.probe-input` angeschlossen und evtl. mit dem Schwimmerschalter

verbunden.

Wie man sieht, sind Plasma-Tische sehr Pin-intensiv, und wir haben bereits etwa 15 Eingänge verbraucht, bevor der normale Notaus-Schalter hinzugefügt wird. Andere haben andere Ansichten, aber der Autor ist der Meinung, dass die Mesa 7I76E der billigeren 7I96 vorzuziehen ist, um MPGs, Skalen- und Achsenwahlschalter und andere Funktionen zu ermöglichen, die Sie vielleicht im Laufe der Zeit hinzufügen möchten. Wenn Ihr Tisch Servos verwendet, gibt es eine Reihe von Alternativen. Es gibt zwar auch andere Anbieter, aber wenn Sie Ihre Maschine um das Mesa-Ökosystem herum konstruieren, wird die Verwendung ihrer THCAD-Platine zum Lesen der Lichtbogenspannung vereinfacht.

Brenner-Abreißsensor (engl. torch breakaway sensor)

- Wie bereits erwähnt, sollte ein Abreißsensor installiert werden, der ausgelöst wird, wenn die Brenner irgendwo gegenläuft und herunterfällt.
- Normalerweise wird dies mit `halui.program-pause` verbunden, damit der Fehler behoben und das Programm fortgesetzt werden kann.

2.7.15. G-Code für Plasmasteuerungen

Die meisten Plasmasteuerungen bieten eine Methode zur Änderung von Einstellungen über G-Code. LinuxCNC unterstützt dies über M67/M68 für analoge Befehle und M62-M65 für digitale (Ein/Aus-Befehle). Wie dies implementiert wird, ist völlig willkürlich. Schauen wir uns an, wie die LinuxCNC QtPlasmaC Konfiguration dies tut:

Wählen Sie die Materialeinstellungen in QtPlasmaC und verwenden Sie die Vorschubgeschwindigkeit für dieses Material.

```
M190 Pn
M66 P3 L3 Q1
F#<_hal[plasmac.cut-feed-rate]>
M3 S1
```

NOTE Benutzer mit einer sehr großen Anzahl von Einträgen in der QtPlasmaC-Materialtabelle müssen möglicherweise den Q-Parameter (z.B. von Q1 auf Q2) erhöhen.

Aktivieren/Deaktivieren des THC-Betriebs:

```
M62 P2 deaktiviert die THC (synchronisiert mit der Bewegung)
M63 P2 aktiviert die THC (synchron mit der Bewegung)
M64 P2 deaktiviert die THC (sofort)
M65 P2 aktiviert THC (sofort)
```

Schnittgeschwindigkeiten reduzieren: (z.B. zum Lochschneiden)

```
M67 E3 Q0 würde die Geschwindigkeit auf 100% der angeforderten~Geschwindigkeit setzen
M67 E3 Q40 würde die Geschwindigkeit auf 40% der angeforderten~Geschwindigkeit setzen
M67 E3 Q60 würde die Geschwindigkeit auf 60% der angeforderten~Geschwindigkeit setzen
M67 E3 Q100 würde die Geschwindigkeit auf 100% der angeforderten~Geschwindigkeit setzen
```

Fräserkompensation:

```
G41.1 D#<_hal[plasmac_run.kerf-width-f]> ; für links von der programmierten Bahn
G42.1 D#<_hal[plasmac_run.kerf-width-f]> für rechts von der programmierten Bahn
G40 zum Ausschalten der Kompensation
```

NOTE Integratoren sollten sich mit der LinuxCNC-Dokumentation für die verschiedenen oben erwähnten LinuxCNC-G-Code-Befehle vertraut machen.

2.7.16. Externe Offsets und Plasmaschneiden

Externe Offsets wurden in LinuxCNC mit Version 2.8 eingeführt. Durch externe, bedeutet es, dass wir einen Offset außerhalb des G-Codes, dass die Flugbahn Planer weiß nichts über anwenden können. Es ist am einfachsten, mit einem Beispiel zu erklären. Stellen Sie sich eine Drehmaschine mit einem externen Offset vor, der durch eine mathematische Formel angewendet wird, um eine Nocke auf einer Kurve zu bearbeiten. Die Drehmaschine dreht sich also blind herum, wobei der Schnittdurchmesser auf einen festen Durchmesser eingestellt ist, und die externe Verschiebung bewegt das Werkzeug nach innen und außen, um die Nocke über eine angewandte externe Verschiebung zu bearbeiten. Um unsere Drehmaschine so zu konfigurieren, dass sie diese Nocke bearbeitet, müssen wir einen Teil der Achsengeschwindigkeit und -beschleunigung den externen Offsets zuweisen, sonst kann sich das Werkzeug nicht bewegen. An dieser Stelle kommt die INI-Variable OFFSET_AV_RATIO ins Spiel. Angenommen, wir entscheiden, dass wir 20 % der Geschwindigkeit und Beschleunigung dem externen Offset der Z-Achse zuweisen müssen. Wir setzen diesen Wert auf 0,2. Dies hat zur Folge, dass die maximale Geschwindigkeit und Beschleunigung für die Z-Achse der Drehmaschine nur 80 % des möglichen Wertes beträgt.

Externe Offsets sind eine sehr leistungsfähige Methode, um die Brennerhöhe über ein THC an der Z-Achse anzupassen. Aber beim Plasma dreht sich alles um hohe Geschwindigkeiten und schnelle Beschleunigungen, so dass es keinen Sinn macht, diese Parameter zu begrenzen. Glücklicherweise wird in einer Plasmamaschine die Z-Achse entweder zu 100% vom THC kontrolliert oder nicht. Während der Entwicklung der externen Offsets von LinuxCNC wurde erkannt, dass sich die Bewegung der Z-Achse durch G-Code und THC gegenseitig ausschließt. Dies ermöglicht es uns, externe Offsets so auszutricksen, dass sie jederzeit 100 % der Geschwindigkeit und Beschleunigung ergeben. Wir können dies tun, indem wir die Geschwindigkeits- und Beschleunigungseinstellungen der Z-Achse der Maschine in der INI-Datei verdoppeln und OFFSET_AV_RATIO = 0,5 einstellen. Auf diese Weise stehen 100% der maximalen Geschwindigkeit und Beschleunigung sowohl für Sondierung als auch für THC zur Verfügung.

Beispiel: An einer metrischen Maschine mit einem NEMA23-Motor mit Direktantrieb auf einen 5 mm-Kugelgewindetrieb wurden 60 mm/s Maximalgeschwindigkeit und 700 mm/s^2 Beschleunigung als sichere Werte ohne Schrittverlust ermittelt. Stellen Sie für diese Maschine die Z-Achse in der INI-Datei wie folgt ein:

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.5
MAX_VELOCITY = 120
MAX_ACCELERATION = 1400
```

Für das mit dieser Achse verbundene Gelenk würden die Geschwindigkeits- und

Beschleunigungsvariablen wie folgt festgelegt:

```
[JOINT_n]
MAX_VELOCITY = 60
MAX_ACCELERATION = 700
```

Für weitere Informationen über externe Offsets (für Version 2.8 oder später) lesen Sie bitte die [AXIS_<letter>] Abschnitt der Beschreibung der INI-Datei und Externe Achsen Offsets in der LinuxCNC Dokumentation.

2.7.17. Messen der Lichtbogenspannung (engl. arc voltage) mit dem Mesa THCAD

Das Mesa THCAD-Board ist ein bemerkenswert preisgünstiger und präziser Spannungs-Frequenz-Wandler, der für die feindliche, laute elektrische Umgebung im Zusammenhang mit Plasmaschneiden ausgelegt ist. Intern hat es einen Bereich von 0-10 V. Dieser Bereich kann einfach durch Hinzufügen einiger Widerstände erweitert werden, wie in der Dokumentation beschrieben. Dieses Board ist in drei Versionen erhältlich, das neuere THCAD-5 mit einem Bereich von 0–5 V, das THCAD-10 mit einem Bereich von 0–10 Volt und das THCAD-300, das für einen erweiterten Bereich von 300 Volt vorkalibriert ist. Jede Platine wird einzeln kalibriert und auf der Platine ist ein Aufkleber angebracht, der die Frequenz bei 0 Volt und Vollausschlag angibt. Für die Verwendung mit LinuxCNC wird empfohlen, den 1/32-Teiler über den entsprechenden Link auf der Platine auszuwählen. Stellen Sie in diesem Fall sicher, dass Sie die angegebenen Frequenzen auch durch 32 teilen. Dies ist besser für den 1 kHz-Servo-Thread geeignet und lässt dem THCAD auch mehr Zeit, um die Ausgabe zu mitteln und zu glätten.

Es gibt eine Menge Verwirrung darüber, wie die THCAD-Ausgabe zu dekodieren ist. Betrachten wir also das Mesa 7I76E und das THCAD-10 für einen Moment mit den folgenden hypothetischen Kalibrierungsdaten:

- Skalenendwert 928 kHz (928 kHz/32 = 29 kHz)
- 0 V 121,6 kHz (121,6 kHz/32 = 3,8 kHz)

Da der Skalenendwert 10 Volt beträgt, ist die Frequenz pro Volt:

(29000 Hz - 3800 Hz) / 10 V = 2520 Hz per Volt

Angenommen, wir haben einen 5 Volt Eingang, so wäre die berechnete Frequenz:

(2520 Hz/V * 5 V) + 3800 Hz = 16400 Hz

Jetzt sollte klar sein, wie man die Frequenz in die entsprechende Spannung umwandelt:

Spannung = (Frequenz [Hz] - 3800 Hz) / (2520 Hz/V)

THCAD Verbindungen

Auf der Hochspannungsseite:

- Verbinden Sie die geteilte oder rohe Lichtbogenspannung mit I_N+ und I_N-

- Verbinden Sie die Abschirmung des Verbindungskabels mit dem Schirmanschluss.
- Verbinden Sie die andere Abschirmungsklemme mit der Rahmenmasse.

Angenommen, er ist an ein Mesa 7I76E angeschlossen, verbinden Sie den Ausgang mit dem Spindelgebereingang:

- THCAD +5 V an TB3 Pin 6 (+5 VP)
- THCAD -5 V an TB3 Pin 1 (GND)
- THCAD FOUT+ an TB3 Pin 7 (ENC A+)
- THCAD FOUT- an TB3 Pin 8 (ENC A-)

THCAD-Erstprüfung

Vergewissern Sie sich, dass Sie die folgenden Zeilen in Ihrer INI-Datei haben (vorausgesetzt, Sie haben eine Mesa 7I76E):

```
setp hm2_7i76e.0.encoder.00.scale -1
setp hm2_7i76e.0.encoder.00.counter-mode 1
```

Schalten Sie Ihren Controller ein und öffnen Sie Halshow (AXIS: Show Homing Configuration), suchen Sie den `hm2_7i76e.0.encoder.00.velocity pin`. Bei angelegten 0 Volt sollte er um die 0-Volt-Frequenz (3.800 in unserem Beispiel) schwanken. Nehmen Sie eine 9-Volt-Batterie und schließen Sie sie an I_N+ und I_N- an. Für ein THCAD-10 können Sie nun die erwartete Geschwindigkeit berechnen (26.480 in unserem hypothetischen Beispiel). Wenn Sie diesen Test bestehen, dann sind Sie bereit, Ihre LinuxCNC Plasmasteuerung zu konfigurieren.

Welches THCAD-Modell soll verwendet werden?

Der THCAD-5 ist nützlich, wenn Sie beabsichtigen, ihn für die Ohmsche Messung zu verwenden. Zweifellos ist das THCAD-10 das flexiblere Gerät und die Skalierung lässt sich leicht ändern. Es gibt jedoch eine Einschränkung, die bei einigen billigeren Plasmaschneidern mit eingebautem Spannungsteiler ins Spiel kommen kann. Das heißt, die internen Widerstände können vom THCAD als Teil seines eigenen externen Widerstands erfasst werden und fehlerhafte Ergebnisse liefern. Zum Beispiel muss der 16:1-Teiler auf den Plasmaschneidern von Everlast als 24:1 behandelt werden (und 50:1 wird 75:1). Bei namhafteren Marken (z. B. Thermal Dynamics, Hypertherm, ESAB usw.) ist dies kein Problem. Wenn Sie also niedrigere Schneidspannungen als erwartet sehen, ist es möglicherweise vorzuziehen, den THCAD neu zu konfigurieren, um die rohe Lichtbogenspannung zu lesen.

Da Plasmalichtbogenspannungen potenziell tödlich sind, werden hier einige Kriterien vorgeschlagen.

Pilotbogen Start

Da keine nennenswerten elektromagnetischen Störungen zu erwarten sind, können Sie den THCAD sicher in Ihrem Schaltschrank installieren, wenn Sie unsere Konstruktionsrichtlinien befolgen.

- Wenn Sie keinen Spannungsteiler haben, installieren Sie entweder Skalierungswiderstände im Plasmaschneider und installieren Sie den THCAD im Bedienfeld oder folgen Sie den Vorschlägen für

HF-Startmaschinen.

- Wenn Sie einen Spannungsteiler haben, installieren Sie einen THCAD-10 in Ihrem Schaltschrank. Wir hatten keine Probleme mit dieser Konfiguration bei einem 120-A-Plasmaschneider von Thermal Dynamics.

HF Start

Installieren Sie den THCAD am Wechselrichter, da das Frequenzsignal weitaus unempfindlicher gegenüber EMI-Rauschen ist.

- Wenn Sie keinen Spannungsteiler haben und im Plasmaschneider Platz haben, installieren Sie ein THCAD-300 im Plasmaschneider.
- Wenn Sie keinen Spannungsteiler haben und innerhalb des Plasmaschneidgeräts keinen Platz haben, installieren Sie ein THCAD-10 in einem Metallgehäuse außerhalb des Plasmaschneidgeräts und installieren Sie jeweils 50% des Skalierungswiderstands an I_N+ und I_N- innerhalb des Plasmaschneidgeräts, damit keine tödlichen Spannungen aus dem Gehäuse kommen.
- Wenn Sie einen Spannungsteiler haben, installieren Sie einen THCAD-10 in einem Metallgehäuse außerhalb des Plasmaschneiders.

Rohspannung des Lichtbogens an einem Stecker

In diesem Fall sind unabhängig von der Lichtbogenstartmethode wahrscheinlich bereits Widerstände in der Schaltung enthalten, um tödliche Schocks zu vermeiden, daher wird ein THCAD-10 empfohlen, damit dieser Widerstand (normalerweise 200 kΩ) bei der Auswahl eines solchen Skalierungswiderstands berücksichtigt werden kann Widerstände verzerren die vom THCAD-300 gemeldete Spannung.

2.7.18. Postprozessoren und Verschachtelung

Plasma unterscheidet sich insofern nicht von anderen CNC-Bearbeitungen, als es sich um eine solche handelt:

1. In CAD entworfen (wo es als DXF- oder manchmal SVG-Format ausgegeben wird).
2. Verarbeitet in CAM, um den endgültigen G-Code zu erzeugen, der in die Maschine geladen wird
3. Schneiden der Teile über CNC-G-Code-Befehle.

Einige Leute erzielen gute Ergebnisse mit Inkscape und G-Code-Tools, aber SheetCam ist eine sehr preisgünstige Lösung und es gibt eine Reihe von Postprozessoren für LinuxCNC. SheetCam verfügt über eine Reihe von erweiterten Funktionen für das Plasmaschneiden und für den Preis ist es ein Kinderspiel für jeden, der regelmäßig plasmaschneidet.

2.7.19. Design für Umgebungen mit Elektrosmog

Plasmaschneiden ist von Natur aus eine extrem feindliche und laute elektrische Umgebung. Wenn Sie EMI-Probleme haben, werden die Dinge nicht richtig funktionieren. Ein offensichtliches Beispiel ist, dass der Brenner gezündet wird und der Computer neu startet, aber es kann auch eine Reihe anderer seltsamer Symptome geben. Sie treten fast alle nur auf, wenn der Brenner schneidet - oft beim ersten Zünden.

Daher sollten Systementwickler die Komponenten sorgfältig auswählen und von Grund auf so konzipieren, dass sie mit dieser feindlichen Umgebung zurechtkommen, um die Auswirkungen elektromagnetischer Störungen (EMI) zu vermeiden. Wird dies nicht beachtet, kann dies zu unzähligen Stunden vergeblicher Fehlersuche führen.

Die Wahl von Ethernet-Karten wie der Mesa 7I76E oder der preiswerteren 7I96 ist hilfreich, da sie es ermöglichen, den PC von der Elektronik und der Plasmamaschine entfernt aufzustellen. Diese Hardware ermöglicht auch die Verwendung von 24-Volt-Logiksystemen, die wesentlich störungsresistenter sind. Die Komponenten sollten in einem Metallgehäuse montiert werden, das mit der Netzerde verbunden ist. Es wird dringend empfohlen, einen EMI-Filter am Netzanschluss zu installieren. Am einfachsten ist es, einen EMI-gefilterten IEC-Netzanschluss zu verwenden, der üblicherweise bei PCs und Elektrogeräten verwendet wird und dies ohne zusätzlichen Aufwand ermöglicht. Planen Sie die Anordnung der Komponenten im Gehäuse so, dass Netzstrom, Hochspannungsmotorleitungen und Logiksignale so weit wie möglich voneinander getrennt sind. Wenn sie sich kreuzen müssen, halten Sie sie in einem Winkel von 90 Grad.

Peter Wallace von Mesa Electronics schlägt vor: "Wenn Sie eine CNC-kompatible Plasmaquelle mit einem Spannungsteiler haben, würde ich den THCAD innerhalb Ihres Elektronikgehäuses zusammen mit der gesamten anderen Bewegungshardware montieren. Wenn Sie eine manuelle Plasmaquelle haben und die Rohspannung des Plasmas ablesen, würde ich den THCAD so nah wie möglich an der Plasmaquelle montieren (sogar im Gehäuse der Plasmaquelle, wenn es passt). Wenn Sie ein abgeschirmtes Gehäuse für die THCAD verwenden, sollte die Abschirmung mit der Masse des Elektronikgehäuses verbunden sein, nicht mit der Masse der Plasmaquelle."

Es wird empfohlen, ein separates Erdungskabel von den Motorgehäusen und dem Brenner zurück zu einem zentralen Sternpunkt an der Maschine zu führen. Verbinden Sie das Plasmaerdungskabel mit diesem Punkt und optional mit einem Erdungsstab, der so nah wie möglich an der Maschine in den Boden getrieben wird (insbesondere bei einer HF-Start-Plasmamaschine).

Die externe Verdrahtung der Motoren sollte abgeschirmt und für den durch den Stromkreis fließenden Strom ausreichend dimensioniert sein. Die Abschirmung sollte auf der Motorseite unverbunden bleiben und auf der Seite des Schaltkastens geerdet werden. Erwägen Sie die Verwendung eines zusätzlichen Pins an allen Anschlüssen des Schaltkastens, so dass die Erdung durch den Schaltkasten hindurch und direkt an der Schritt-/Servomotorsteuerung selbst mit dem Gehäuse geerdet werden kann.

Uns ist mindestens ein kommerzieller Systembauer bekannt, der Probleme mit induziertem elektrischem Rauschen im ohmschen Messkreis hatte. Dies kann zwar durch die Verwendung von Ferritperlen und das Aufwickeln des Kabels gemildert werden, aber es wird auch empfohlen, an der Stelle, an der das ohmsche Messsignal in das Elektronikgehäuse eintritt, einen Netzfilter einzusetzen.

Tommy Berisha, der Meister im Bau von Plasmageräten mit kleinem Budget, sagt: "Wenn Sie ein kleines Budget haben, sollten Sie alte Laptop-Netzteile verwenden. Sie sind sehr gut, filtern gut, sind komplett isoliert, strombegrenzt (das wird sehr wichtig, wenn etwas schief geht), und es ist einfach, 2 oder 3 von ihnen in Reihe zu schalten, da sie isoliert sind. Beachten Sie, dass bei einigen die Erdung mit dem negativen Ausgangskontakt verbunden ist, sie muss also abgeklemmt werden, was einfach durch die Verwendung eines Stromkabels ohne Erdungskontakte möglich ist)."

2.7.20. Wasser-Tische

Der minimale Wasserstand unter der Schnittebene des Brenners sollte etwa 40 mm betragen. Es ist gut, wenn unter den Lamellen Platz ist, damit das Wasser während des Schneidens ablaufen und entweichen kann; ein wenig Wasser über der zu schneidenden Metallplatte ist wirklich gut, da es das bisschen Staub beseitigt. Die Zugabe von Backpulver zum Wasser hält den Tisch für viele Jahre in einem guten Zustand, da es keine Korrosion zulässt, während die Lamellen unter Wasser sind, und es reduziert auch den Geruch des Wasserdampfes. Manche Leute verwenden ein Wasserreservoir mit einem Drucklufteinlass, um das Wasser aus dem Reservoir bei Bedarf auf den Wassertisch zu drücken und so Änderungen des Wasserstands zu ermöglichen.

2.7.21. Downdraft-Tische

Viele handelsübliche Tische sind nach unten gezogen, d. h., es werden Ventilatoren eingesetzt, welche die Luft durch die Lamellen nach unten saugen, um Dämpfe und Funken aufzufangen. Oft sind die Tische in Zonen eingeteilt, so dass nur ein Abschnitt unterhalb des Brenners für die Abluft geöffnet ist, wobei häufig Luftkolben und Luftmagnetventile zum Öffnen der Klappen verwendet werden. Das Auslösen dieser Zonen ist relativ einfach, wenn Sie die Achsen- oder Gelenkposition von einem der Bewegungspins und die Lincurve-Komponente verwenden, um die Abluftzonen dem richtigen Ausgangspin zuzuordnen.

2.7.22. Design für Geschwindigkeit und Beschleunigung

Beim Plasmaschneiden kommt es auf Geschwindigkeit und Beschleunigung an. Je höher die Beschleunigung, desto weniger muss die Maschine bei Kurvenfahrten abbremsen. Das bedeutet, dass das Portal so leicht wie möglich sein sollte, ohne dabei an Torsionssteifigkeit einzubüßen. Ein 100 mm x 100 mm x 2 mm großes Aluminium-Kastenprofil hat die gleiche Torsionssteifigkeit wie ein 80 mm x 80 mm großes T-Nut-Strangpressprofil, ist aber 62 % leichter. Überwiegen also die Vorteile der T-Nuten die zusätzlichen Konstruktionskosten?

2.7.23. Zurückgelegte Strecke pro Motorumdrehung

Schrittmotoren leiden unter Resonanz und ein direkt angetriebenes Ritzel bedeutet wahrscheinlich, dass der Motor unter ungünstigen Bedingungen arbeitet. Für Plasmamaschinen wird ein Abstand von etwa 15-25 mm pro Motorumdrehung als ideal angesehen, aber auch etwa 30 mm pro Umdrehung sind noch akzeptabel. Eine Kugelumlaufspindel mit 5 mm Steigung und einer Untersetzung von 3:1 oder 5:1 ist ideal für die Z-Achse.

2.7.24. QtPlasmaC LinuxCNC Plasma Konfiguration

QtPlasmaC, der aus einer HAL-Komponente (plasmac.hal) und einer vollständigen Konfiguration für die QtPlasmaC-GUI besteht, hat beträchtlichen Input von vielen in der LinuxCNC-Open-Source-Bewegung erhalten, die das Verständnis von Plasma vorangetrieben haben Controller seit etwa 2015. Es gab viel Test- und Entwicklungsarbeit, um QtPlasmaC in seinen aktuellen Betriebszustand zu bringen. Alles vom Schaltungsdesign bis zur G-Code-Steuerung und Konfiguration ist enthalten. Darüber hinaus unterstützt QtPlasmaC externe THCs wie den Proma 150, kommt aber wirklich zur Geltung, wenn er mit einem Mesa-Controller gekoppelt wird, da dies dem Integrator ermöglicht, den Mesa THCAD-Spannungs-

Frequenz-Wandler einzubeziehen, der speziell für die feindliche Plasmaumgebung entwickelt wurde.

QtPlasmaC ist als eigenständig konzipiert und bietet die Möglichkeit, Ihre Schnittdiagramme einzubeziehen, enthält aber auch Funktionen, die mit einem Postprozessor wie SheetCam verwendet werden können.

Das QtPlasmaC-System ist jetzt in Version 2.9 und höher von LinuxCNC enthalten. Es ist jetzt ziemlich ausgereift und wurde seit der ersten Version dieses Handbuchs erheblich verbessert. QtPlasmaC wird die Plasmaunterstützung von LinuxCNC für viele Jahre definieren, da es alle Funktionen eines proprietären High-End-Plasmasteuerungssystems zu einem Open-Source-Preis enthält.

2.7.25. Hypertherm RS485 Steuerung

Einige Hypertherm-Plasmaschneider haben eine RS485-Schnittstelle, um der Steuerung (z. B. LinuxCNC) zu ermöglichen, Ampere, Druck und Modus einstellen zu können. Einige Leute haben eine in Python geschriebene Nicht-Echtzeit-Komponente verwendet, um dies zu erreichen. Seit kurzem unterstützt QtPlasmaC diese Schnittstelle nun auch nativ. Lesen Sie in der QtPlasmaC-Dokumentation nach, wie Sie es verwenden können.

Die Kombination aus einer langsamen Baudrate, die von Hypertherm verwendet wird, und der Nicht-Echtzeit-Komponente führt dazu, dass sich der Maschinenzustand nur sehr langsam ändert, so dass es im Allgemeinen nicht möglich ist, Einstellungen während des Schneidens zu ändern.

Bei der Auswahl einer RS485-Schnittstelle auf der PC-Seite haben Benutzer berichtet, dass USB-zu-RS485-Schnittstellen nicht zuverlässig sind. Gute und zuverlässige Ergebnisse wurden mit einer hardwarebasierten RS232-Schnittstelle (z. B. PCI/PCIe oder Motherboard-Port) und einem geeigneten RS485-Konverter erzielt. Einige Benutzer haben über Erfolge mit einer Sunix P/N: SER5037A PCI RS2322-Karte und einem generischen XC4136 RS232-zu-RS485-Konverter (der manchmal auch ein USB-Kabel enthält) berichtet.

2.7.26. Postprozessoren für das Plasmaschneiden

CAM-Programme (Computer Aided Manufacture) sind die Brücke zwischen CAD (Computer Aided Design) und der endgültigen CNC-Bearbeitung (Computer Numerical Control). Sie enthalten oft einen vom Benutzer konfigurierbaren Postprozessor, um den Code zu definieren, der für eine bestimmte Maschine oder einen bestimmten Dialekt des G-Codes erzeugt wird.

Viele LinuxCNC-Benutzer sind vollkommen zufrieden mit der Verwendung von Inkscape zu konvertieren SVG vektorbasierten Dateien in G-Code. Wenn Sie mit einem Plasmaschneider für Hobby- oder Heimgebrauch sind, sollten Sie diese Option.

Wenn Ihre Anforderungen jedoch komplexer sind, ist die wahrscheinlich beste und preiswerteste Lösung SheetCam. SheetCam unterstützt sowohl Windows als auch Linux und es sind Postprozessoren dafür verfügbar, einschließlich der QtPlasmaC-Konfiguration. SheetCam ermöglicht das Verschachteln von Teilen über eine ganze Materialplatte und erlaubt die Konfiguration von Toolsets und Codeschnipseln nach Ihren Bedürfnissen. SheetCam-Postprozessoren sind Textdateien, die in der Programmiersprache Lua geschrieben sind und im Allgemeinen leicht an Ihre genauen Anforderungen angepasst werden können. Weitere Informationen finden Sie auf der SheetCam-Website und in deren

Support-Forum.

Ein weiterer beliebter Postprozessor ist in dem beliebten Fusion360-Paket enthalten, aber die enthaltenen Postprozessoren müssen angepasst werden.

LinuxCNC ist eine CNC-Anwendung und Diskussionen über andere CAM-Techniken als diese Einführungsdiskussion liegen außerhalb des Rahmens von LinuxCNC.

[1] Gefunden bei link:http://en.wikipedia.org/wiki/Separation_of_mechanism_and_policy, 2022-11-13

[2] Gefunden auf link:https://en.wikipedia.org/wiki/Unix_philosophy, 07/06/2008

Chapter 3. Konfigurationsassistenten

3.1. Schrittmotor (engl. stepper) Konfigurationsassistent

3.1.1. Einführung

LinuxCNC ist in der Lage, eine breite Palette von Maschinen mit vielen verschiedenen Hardware-Schnittstellen zu steuern.

StepConf ist ein Programm, das Konfigurationsdateien für LinuxCNC für eine bestimmte Klasse von CNC-Maschinen generiert: diejenigen, die über einen *Standard-Parallelport* und durch *Schritt & Richtung* -Signale gesteuert werden.

StepConf wird bei der Installation von LinuxCNC mitinstalliert und befindet sich im CNC-Menü.

StepConf legt eine Datei im Verzeichnis linuxcnc/config ab, um die Auswahlmöglichkeiten für jede von Ihnen erstellte Konfiguration zu speichern. Wenn Sie etwas ändern, müssen Sie die Datei auswählen, die dem Namen Ihrer Konfiguration entspricht. Die Dateierweiterung lautet .stepconf.

Der StepConf-Assistent funktioniert am besten bei einer Bildschirmauflösung von mindestens 800 x 600.

3.1.2. Startseite

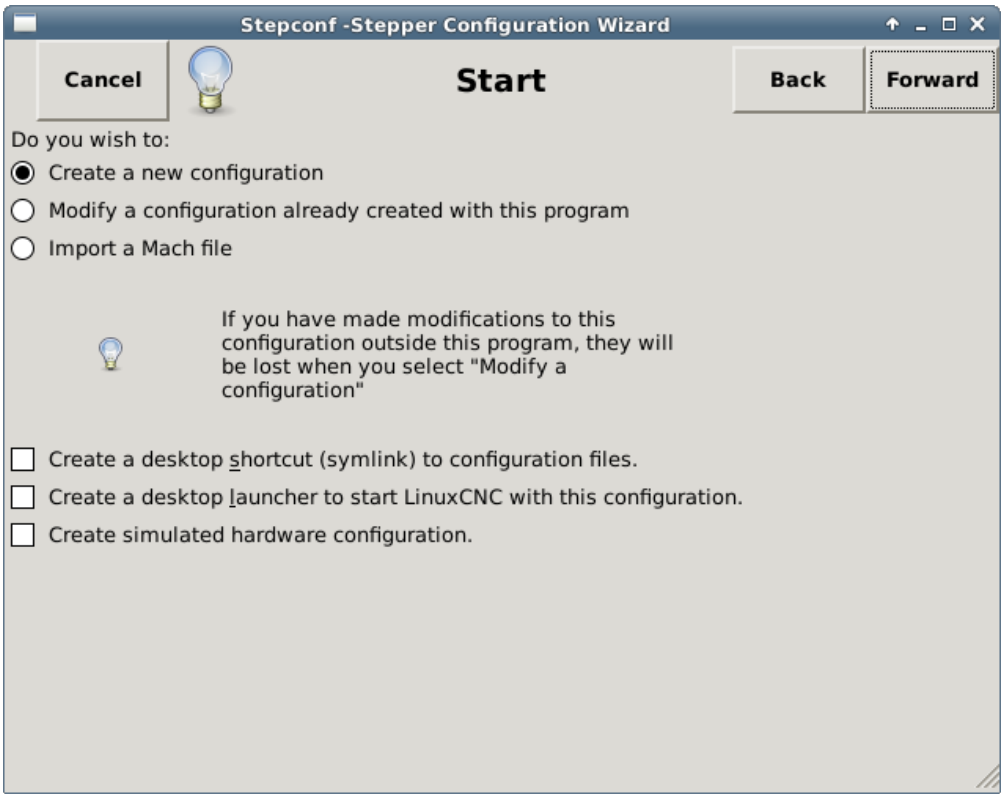


Figure 27. StepConf Einstiegsseite

Die ersten drei Optionsfelder sind selbsterklärend:

- *Neu erstellen* (engl. Create New) - legt eine neue Konfiguration an.

- *Ändern* - (engl. Modify) Ändern Sie eine bestehende Konfiguration. Nachdem Sie dies ausgewählt haben, erscheint eine Dateiauswahl, mit der Sie die zu ändernde .stepconf-Datei auswählen können. Wenn Sie Änderungen an der Haupt (engl. main)-HAL- oder der INI-Datei vorgenommen haben, gehen diese verloren. Änderungen an custom.hal und custom_postgui.hal werden vom StepConf-Assistenten nicht geändert. StepConf markiert die zuletzt erstellte Conf-Datei.
- *Importieren* (engl. import) - Importieren Sie eine Mach-Konfigurationsdatei und versucht, sie in eine LinuxCNC-Konfigurationsdatei zu konvertieren. Nach dem Import gehen Sie durch die Seiten von StepConf, um die Einträge zu bestätigen/zu ändern. Die ursprüngliche Mach-XML-Datei wird nicht verändert.

Diese folgenden Optionen werden in einer Einstellungsdatei für den nächsten Lauf von StepConf gespeichert.

- *Desktop-Verknüpfung erstellen* (engl. Create Desktop Shortcut) - Damit wird eine Verknüpfung auf Ihrem Desktop zu den Dateien erstellt.
- *Desktop Launcher erstellen* (engl. Create Desktop Launcher) - Damit wird ein Launcher auf Ihrem Desktop platziert, um Ihre Anwendung zu starten.
- *Simulierte Hardware erstellen* (engl. Create Simulated Hardware) - Damit können Sie eine Konfiguration zum Testen erstellen, auch wenn Sie nicht über die tatsächliche Hardware verfügen'.

3.1.3. Grundlegende Informationen

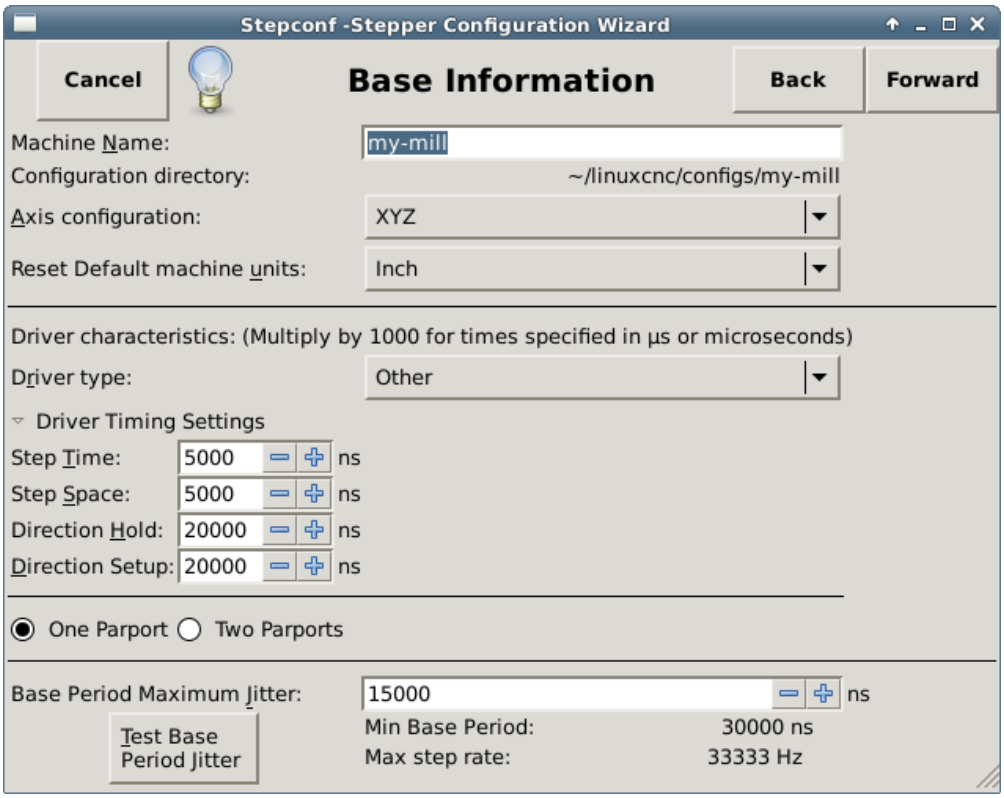


Figure 28. Seite mit grundlegenden Informationen

- *Simulierte Hardware erstellen* (engl. Create Simulated Hardware) - Damit können Sie eine Konfiguration zum Testen erstellen, auch wenn Sie nicht über die tatsächliche Hardware verfügen'.
- *Maschinenname* - Wählen Sie einen Namen für Ihre Maschine. Verwenden Sie nur Großbuchstaben,

Kleinbuchstaben, Ziffern, - und _.

- *Achsenkonfiguration* - Wählen Sie XYZ (Fräsen), XYZA (4-Achsen-Fräsen) oder XZ (Drehen).
- *Maschineneinheiten* (engl. machine units) - Wählen Sie Zoll (engl. inch) oder mm. Alle nachfolgenden Eingaben werden in der gewählten Einheit vorgenommen. Wenn Sie diese Einstellung ändern, werden auch die Standardwerte im Bereich Achsen geändert. Wenn Sie dies ändern, nachdem Sie Werte in einem der Achsenbereiche ausgewählt haben, werden diese durch die Standardwerte der ausgewählten Einheiten überschrieben.
- *Treiber Typ* - Wenn Sie einen der in der Pulldown-Box aufgeführten Schrittmotor (engl. stepper)-Treiber (engl. driver) haben, wählen Sie ihn aus. Andernfalls wählen Sie *Andere* und suchen Sie die Timing-Werte im Datenblatt Ihres Treibers' und geben Sie sie als *Nanosekunden* im Feld *Treiber-Timing-Einstellungen* ein. Wenn im Datenblatt ein Wert in Mikrosekunden angegeben ist, multiplizieren Sie ihn mit 1000. Geben Sie zum Beispiel 4,5 µs als 4500 ns ein.

  Eine Liste einiger beliebter Antriebe, zusammen mit ihren Timing-Werte, ist auf der LinuxCNC.org Wiki unter Stepper Drive Timing.

  Zusätzliche Signalkonditionierung oder -isolierung wie Optokoppler und RC-Filter auf Break-Out-Platinen können zusätzlich zu den Zeitvorgaben des Treibers eigene Einschränkungen mit sich bringen. Es kann notwendig sein, die Treiberanforderungen um einige Zeit zu verlängern, um dies zu berücksichtigen.

  Die LinuxCNC Konfigurations-Auswahl hat Einstellungen für Sherline bereits vorbereitet.

- *Step Time* - Wie lange der Schrittimpuls *on* in Nanosekunden ist. Wenn Sie sich bei dieser Einstellung nicht sicher sind, funktioniert ein Wert von 20.000 bei den meisten Antrieben.
- *Schrittweite* (engl. Step Space) - Minimale Zeit zwischen den Schrittimpulsen in Nanosekunden. Wenn Sie sich bei dieser Einstellung nicht sicher sind, funktioniert ein Wert von 20.000 bei den meisten Antrieben.
- *Direction Hold* - Wie lange der Richtungs-Pin nach einer Richtungsänderung in Nanosekunden gehalten wird. Wenn Sie sich bei dieser Einstellung nicht sicher sind, wird ein Wert von 20.000 bei den meisten Antrieben funktionieren.
- *Direction Setup* - Zeitraum vor einem Richtungswechsel nach dem letzten Schrittimpuls in Nanosekunden. Wenn Sie sich bei dieser Einstellung nicht sicher sind, funktioniert ein Wert von 20.000 bei den meisten Antrieben.
- *One / Two Parport* - Wählen Sie, wie viele parallele Anschlüsse konfiguriert werden sollen.
- *Base Period Maximum Jitter* - Geben Sie hier das Ergebnis des Latenztests ein. Um einen Latenztest durchzuführen, drücken Sie die Schaltfläche *Test Base Period Jitter*. Weitere Einzelheiten finden Sie im Abschnitt Latenz-Test.

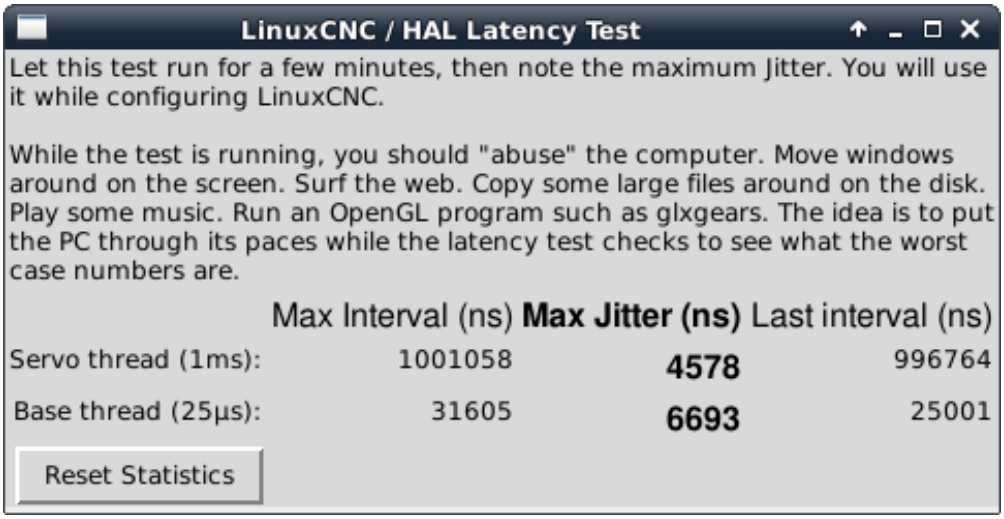


Figure 29. Latenz-Test

- *Max Step Rate* - StepConf berechnet die Maximale Schrittgeschwindigkeit automatisch auf der Grundlage der eingegebenen Treibermerkmale und des Latenz-Test-Ergebnisses.
- *Min Base Period* - StepConf ermittelt die Mininmal Basis Periode automatisch auf der Grundlage der eingegebenen Treibermerkmale und des Latenz-Test-Ergebnisses.

Die wichtige Zahl im Ergebnis des Latenz-Test ist der *maximale Jitter*. Im obigen Beispiel liegt dieser maximal beobachtete Jitter bei 9075 Nanosekunden (ns) oder 9,075 Mikrosekunden (µs). Notieren Sie diese Zahl, und geben Sie sie in das Feld Base Period Maximum Jitter ein.

3.1.4. Einrichtung der parallelen Schnittstelle

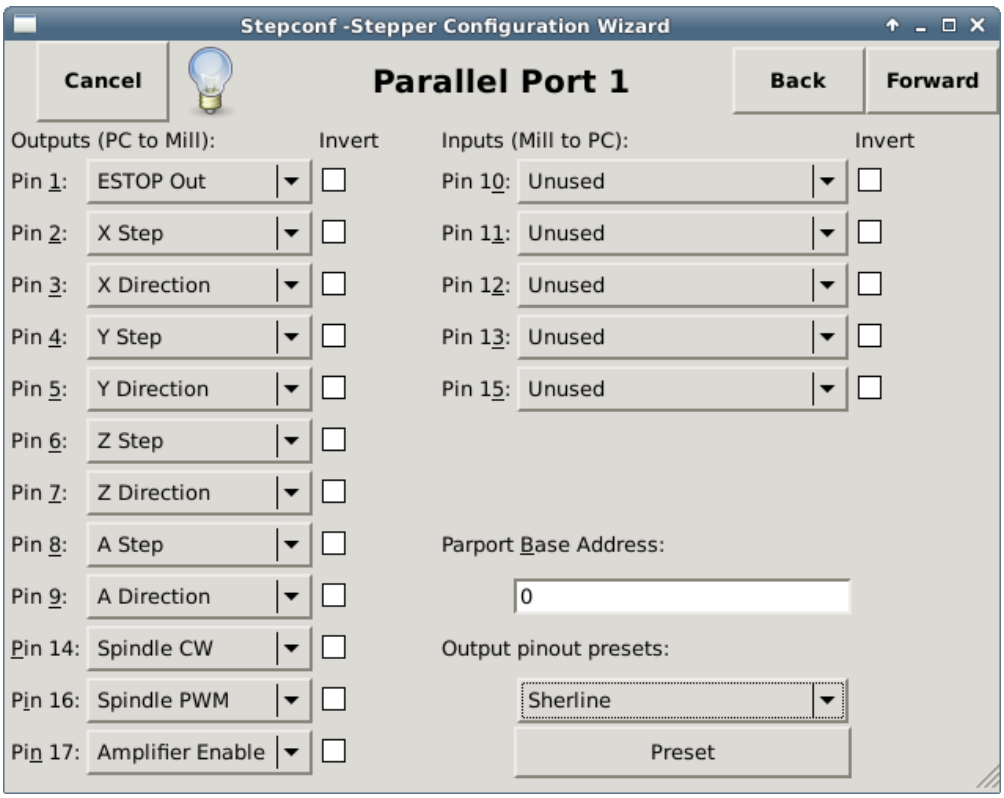


Figure 30. Parallele Schnittstelle Setup-Seite

Sie können die Adresse als Hexadezimalwert (oft 0x378) oder als Linux's Standard-Portnummer

(wahrscheinlich 0) angeben

Wählen Sie für jeden Pin das Signal, das der Pinbelegung Ihres Parallelports entspricht. Aktivieren Sie das Kontrollkästchen *invert*, wenn das Signal invertiert ist (0V für wahr/aktiv, 5V für falsch/inaktiv).

- *Ausgangspinout-Voreinstellungen* - Automatische Einstellung der Pins 2 bis 9 gemäß dem Sherline-Standard (Richtung auf Pins 2, 4, 6, 8) oder dem Xylotex-Standard (Richtung auf Pins 3, 5, 7, 9).
- „Eingänge und Ausgänge“ – Wenn der Ein- oder Ausgang nicht verwendet wird, setzen Sie die Option auf „Nicht verwendet“.
- *Externes Notaus* (engl. external E-Stop) - Dies kann aus einem Dropdown-Feld für den Eingangsstift ausgewählt werden. Eine typische Notaus-Kette verwendet alle normalerweise geschlossenen Kontakte.
- *Referenzpunkt- & Endschalter* - Diese können bei den meisten Konfigurationen aus einem Dropdown-Feld für den Eingangspin ausgewählt werden.
- *Charge Pump* - Wenn Ihre Treiberplatine ein Ladungspumpensignal benötigt, wählen Sie Charge Pump aus der Dropdown-Liste für den Ausgangspin, den Sie mit Ihrem Ladungspumpeneingang verbinden möchten. Der Ausgang der Ladungspumpe ist über StepConf mit dem Basisgewinde verbunden. Der Ausgang der Ladungspumpe entspricht etwa der Hälfte der maximalen Schrittrate, die auf der Seite "Basic Machine Configuration" angegeben ist.
- *Plasmabogen-Spannung* - (engl. Plasma Arc Voltage) Wenn Sie eine Mesa THCAD zur Eingabe einer Plasmalichtbogenspannung benötigen, wählen Sie Plasmabogen-Spannung aus der Liste der Ausgangspins. Dadurch wird während des Setup-Vorgangs eine THCAD-Seite für die Eingabe der Kartenparameter aktiviert.

3.1.5. Einrichtung des zweiten parallelen Ports

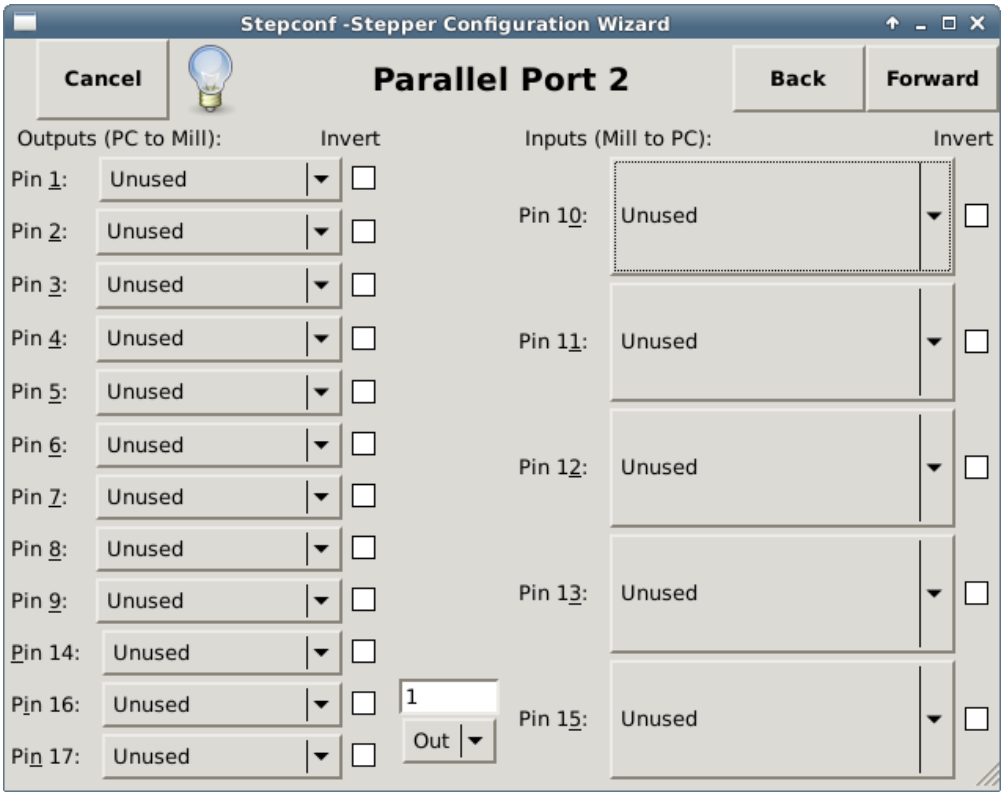


Figure 31. Einrichten von Parallel Port 2

Der zweite Parallelport (falls ausgewählt) kann auf dieser Seite konfiguriert und seine Pins zugewiesen werden. Es können keine Schritt- und Richtungssignale ausgewählt werden. Sie können "in" oder "out" wählen, um die Anzahl der verfügbaren Eingangs-/Ausgangs-Pins zu maximieren. Sie können die Adresse als Hexadezimalwert (oft 0x378) oder als Linux's Standard-Portnummer (wahrscheinlich 1) angeben.

3.1.6. Konfiguration der Achsen

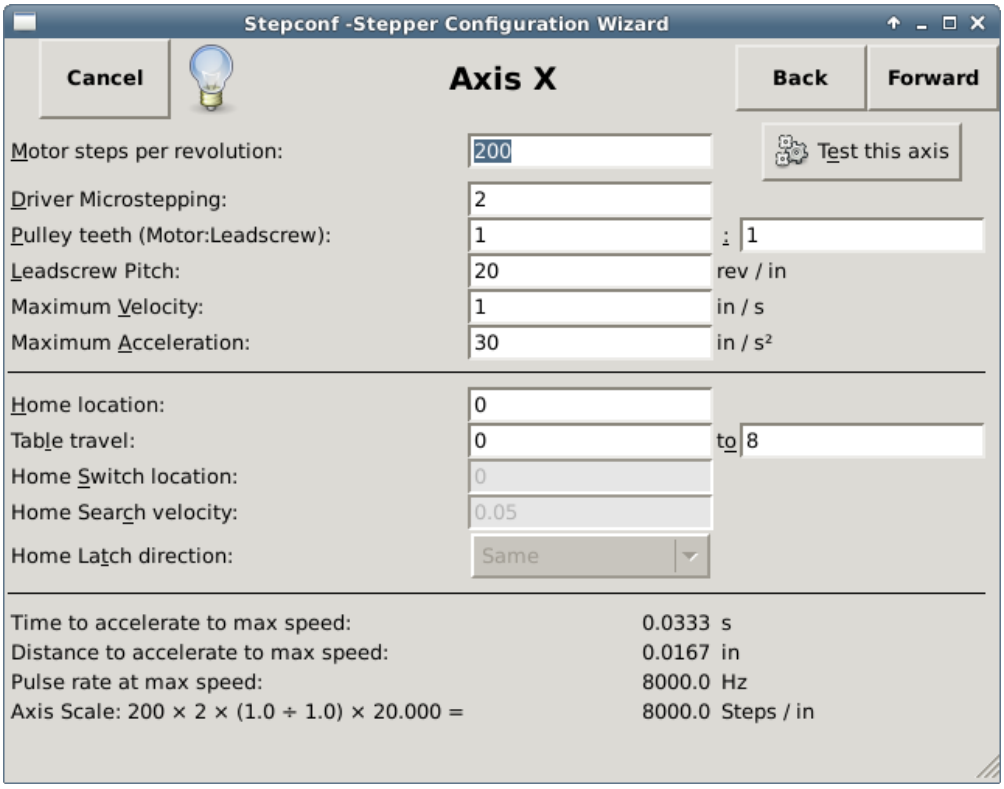


Figure 32. Achsenkonfiguration am Bildschirm

- *Motorschritte pro Umdrehung* - Die Anzahl der vollen Schritte pro Motorumdrehung. Wenn Sie wissen, wie viele Grad pro Schritt der Motor hat (z. B. 1,8 Grad), dann teilen Sie 360 durch die Gradzahl pro Schritt, um die Anzahl der Schritte pro Motorumdrehung zu ermitteln.
- *Microstepping des Treibers* - Der Umfang des vom Treiber durchgeführten Mikroschrittes, engl. driver microstepping. Geben Sie *2* für halbes Stepping ein.
- *Riemenscheibenverhältnis* - Wenn Ihre Maschine Riemenscheiben zwischen Motor und Leitspindel hat, geben Sie hier das Verhältnis ein. Wenn nicht, geben Sie *1:1* ein.
- *Gewindespindelsteigung* - Geben Sie hier die Steigung der Leitspindel ein. Wenn Sie die Einheit *Zoll* gewählt haben, geben Sie die Anzahl der Gewinde pro Zoll ein. Wenn Sie die Einheit *mm* gewählt haben, geben Sie die Anzahl der Millimeter pro Umdrehung ein (z.B. 2 für 2mm/Umdrehung). Wenn die Maschine in die falsche Richtung fährt, geben Sie hier eine negative Zahl anstelle einer positiven Zahl ein, oder kehren Sie den Richtungspin für die Achse um.
- *Maximale Geschwindigkeit* - Geben Sie die maximale Geschwindigkeit für die Achse in Einheiten pro Sekunde ein.
- *Maximale Beschleunigung* - Die richtigen Werte für diese Elemente können nur durch Experimentieren ermittelt werden. Siehe Maximale Geschwindigkeit bestimmen zur Einstellung der Geschwindigkeit und Maximale Beschleunigung bestimmen zur Einstellung der Beschleunigung.
- *Referenzpunkt* - Die Position, zu der die Maschine nach Abschluss des Referenzfahrtverfahrens für diese Achse fährt (engl. home location). Bei Maschinen ohne Referenzfahrtschalter ist dies die Position, zu der ein Bediener die Maschine manuell bewegt, bevor er die Taste Home drückt. Wenn Sie den Referenzpunktschalter und den Endschalter kombinieren, müssen Sie den Schalter in die Referenzpunktposition fahren, da sonst ein Fehler in der Gelenkbegrenzung auftritt.
- *Tischverfahrweg* - Der Verfahrbereich für diese Achse, bezogen auf den Maschinenursprung, engl. table travel. Die Ausgangsposition muss innerhalb des *Tischverfahrwegs* liegen und darf nicht gleich

einem der Werte des Tabellenverfahrwegs sein.

- *Referenzfahrtschalter-Position* - (engl. Home Switch Location) Die Position, an der ein Home-Schalter ausgelöst oder freigegeben wird, bezogen auf den Ursprung der Maschine. Dieser Punkt und die beiden folgenden erscheinen nur, wenn in der Pinbelegung des parallelen Anschlusses Referenzfahrtschalter (engl. home switches))) gewählt wurde. Wenn Sie Referenzfahrt- und Endschalter kombinieren, darf die Position des Referenzfahrtschalters nicht mit der Referenzfahrtposition übereinstimmen, da sonst ein gemeinsamer Endschalterfehler auftritt.
- *Referenzpunkt Such-Geschwindigkeit* (engl. Home Search Velocity) - Die Geschwindigkeit, die bei der Suche nach dem Referenz-Schalter verwendet wird. Befindet sich der Schalter in der Nähe des Endes des technisch möglichen Verfahrwegs, muss diese Geschwindigkeit so gewählt werden, dass die Achse vor dem Erreichen des Endes des Verfahrwegs bis zum Stillstand abbremsen kann. Wenn der Schalter nur für einen kurzen Bereich des Verfahrwegs geschlossen ist (anstatt von seinem Auslösepunkt bis zu einem Ende des Verfahrwegs geschlossen zu sein), muss diese Geschwindigkeit so gewählt werden, dass die Achse bis zum Anschlag abbremsen kann, bevor der Schalter wieder geöffnet wird, und die Referenzfahrt muss immer von derselben Seite des Schalters aus gestartet werden. Wenn sich die Maschine zu Beginn der Referenzfahrt in die falsche Richtung bewegt, negieren Sie diesn Wert.
- *Home Latch Direction* - Wählen Sie *Same*, damit die Achse vom Schalter zurückfährt und sich ihm dann mit sehr geringer Geschwindigkeit wieder nähert. Wenn sich der Schalter zum zweiten Mal schließt, wird die Grundstellung eingestellt. Wählen Sie *Umgekehrt*, damit die Achse vom Schalter zurückfährt und beim Öffnen des Schalters der Referenzpunkt eingestellt wird.
- *Zeit bis zum Erreichen der Maximalgeschwindigkeit* - (engl. time to accelerate to max speed) Zeit bis zum Erreichen der Höchstgeschwindigkeit, berechnet aus *maximaler Beschleunigung* (engl. max acceleration) und *maximaler Geschwindigkeit* (engl. max velocity).
- *Entfernung zur Beschleunigung auf Höchstgeschwindigkeit* - (engl. distance to accelerate to max speed) Entfernung zum Erreichen der Höchstgeschwindigkeit aus dem Stand.
- *Pulsfrequenz bei maximaler Geschwindigkeit* (engl. pulse rate at max speed) - Informationen, die auf der Grundlage der oben eingegebenen Werte berechnet werden. Die größte *Impulsrate bei maximaler Geschwindigkeit* bestimmt die *BASIS_PERIOD*. Werte über 20000Hz können zu langsamen Reaktionszeiten oder sogar zu Blockierungen führen (die schnellste nutzbare Pulsrate variiert von Computer zu Computer)
- *Axis SCALE* - Die Zahl, die in der INI-Datei [SCALE] Einstellung verwendet wird. Die Anzahl Schritte pro Benutzereinheit.
- *Diese Achse testen* - (engl. test this axis) Dadurch wird ein Fenster geöffnet, in dem jede Achse getestet werden kann. Dies kann verwendet werden, nachdem alle Informationen für diese Achse ausgefüllt wurden.

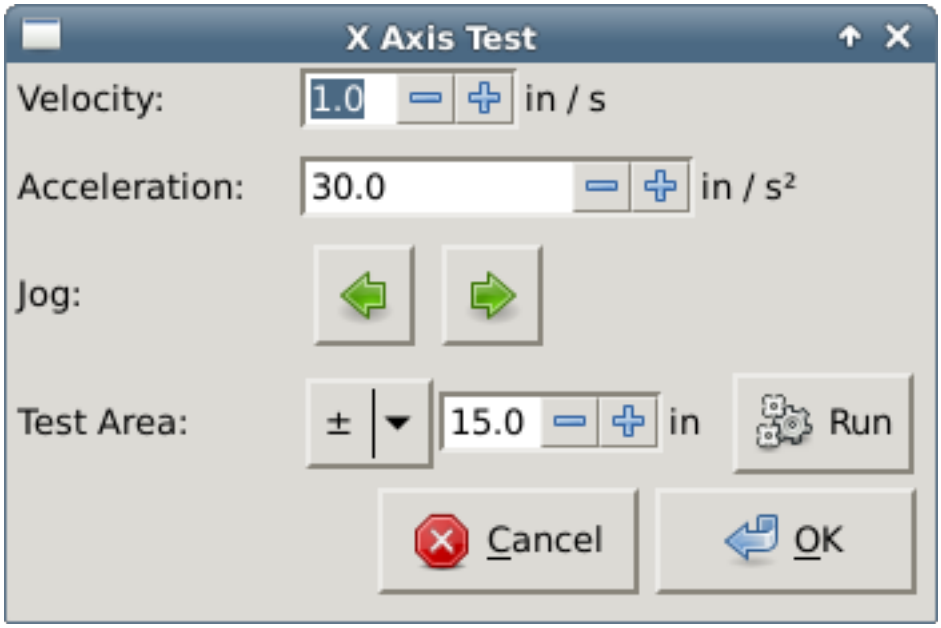


Figure 33. Achsen-Test

Der Achsen-Test ist ein einfacher Test, für den nur Schritt- und Richtungssignale ausgegeben werden, um verschiedene Werte für Beschleunigung und Geschwindigkeit zu testen.

IMPORTANT

Um diese Achse zu testen, müssen Sie die Achse manuell aktivieren, wenn dies erforderlich ist. Wenn Ihr Treiber über eine Ladungspumpe verfügt, müssen Sie diese überbrücken. Der Test dieser Achse reagiert nicht auf Endschaltereingänge. Verwenden Sie diesen mit Vorsicht.

Bestimmen der maximalen Geschwindigkeit

Beginnen Sie mit einer geringen Beschleunigung (z. B. `2 Zoll/s²` oder `50 mm/s²`) und die Geschwindigkeit, die Sie zu erreichen hoffen. Verwenden Sie die Tasten, um die Achse in die Nähe der Mitte des Verfahrwegs zu bewegen. Seien Sie vorsichtig, denn bei einem niedrigen Beschleunigungswert kann es überraschend lange dauern, bis die Achse bis zum Stillstand abbremst.

Nachdem Sie den verfügbaren Verfahrweg gemessen haben, geben Sie im Testbereich eine sichere Entfernung ein, wobei Sie bedenken müssen, dass sich der Motor nach dem Abwürgen in eine unerwartete Richtung bewegen kann. Klicken Sie dann auf Ausführen. Die Maschine beginnt nun, sich entlang dieser Achse hin und her zu bewegen. Bei diesem Test ist es wichtig, dass die Kombination aus Beschleunigung und Testbereich es der Maschine ermöglicht, die gewählte Geschwindigkeit zu erreichen und zumindest eine kurze Strecke zu fahren - je mehr Strecke, desto besser ist dieser Test. Die Formel `d = 0,5 * v * v/a` gibt den Mindestabstand an, der erforderlich ist, um die angegebene Geschwindigkeit mit der gegebenen Beschleunigung zu erreichen. Wenn es bequem und sicher ist, schieben Sie den Tisch gegen die Bewegungsrichtung, um Schnittkräfte zu simulieren. Wenn die Maschine zum Stillstand kommt, verringern Sie die Geschwindigkeit und starten Sie den Test erneut.

Wenn die Maschine nicht offensichtlich zum Stillstand gekommen ist, klicken Sie auf die Schaltfläche *Ausführen*. Die Achse kehrt nun zu der Position zurück, an der sie gestartet ist. Wenn die Position nicht korrekt ist, dann ist die Achse während des Tests stehen geblieben oder hat Schritte verloren. Verringern Sie die Geschwindigkeit und starten Sie den Test erneut.

Wenn sich die Maschine nicht bewegt, stehen bleibt oder Schritte verliert, egal wie niedrig Sie die Geschwindigkeit einstellen, überprüfen Sie Folgendes:

- Korrigieren Sie das Timing der Schrittwellenform
- Korrekte Pinbelegung, einschließlich *Invert* auf Step-Pins
- Korrekte, gut geschirmte Verkabelung
- Physikalische Probleme mit dem Motor, der Motorkupplung, der Leitspindel usw.

Sobald Sie eine Geschwindigkeit gefunden haben, bei der die Achse während dieser Testprozedur nicht ins Stocken gerät oder Schritte verliert, reduzieren Sie diese um 10 % und verwenden Sie diese Geschwindigkeit als Maximalgeschwindigkeit der Achse.

Bestimmen der maximalen Beschleunigung

Geben Sie mit der im vorherigen Schritt ermittelten Höchstgeschwindigkeit den zu testenden Beschleunigungswert ein. Passen Sie den Beschleunigungswert wie oben beschrieben nach oben oder unten an. Bei diesem Test ist es wichtig, dass die Kombination aus Beschleunigung und Testbereich es der Maschine ermöglicht, die ausgewählte Geschwindigkeit zu erreichen. Sobald Sie einen Wert gefunden haben, bei dem die Achse während dieses Testverfahrens nicht ins Stocken gerät oder Schritte verliert, reduzieren Sie ihn um 10 % und verwenden Sie diesen Wert als maximale Beschleunigung der Achse.

3.1.7. Spindel-Konfiguration

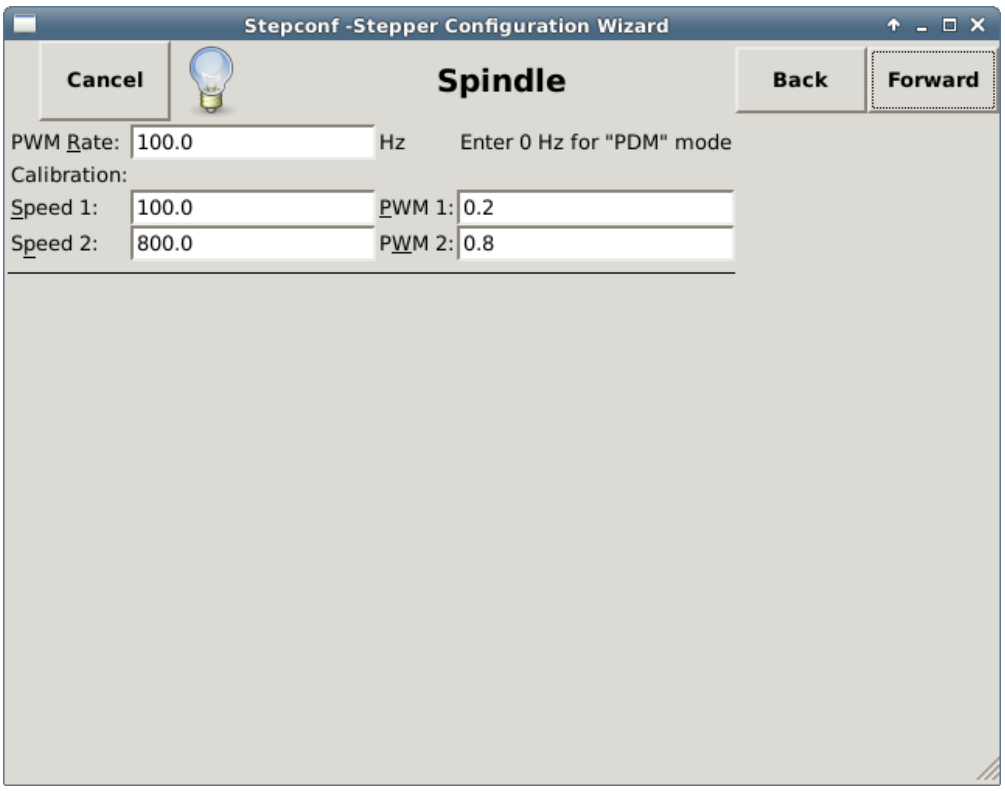


Figure 34. Seite zur Spindelkonfiguration

Diese Seite erscheint nur, wenn *Spindle PWM* auf der Seite *Parallel Port Pinout* für einen der Ausgänge

gewählt wurde.

Steuerung der Spindelgeschwindigkeit

Wenn *Spindle PWM* auf dem Pinout erscheint, sollten die folgenden Informationen eingegeben werden:

- *PWM Rate* - Die *Trägerfrequenz* des PWM-Signals für die Spindel. Geben Sie *0* für den PDM-Modus ein, der für die Erzeugung einer analogen Steuerspannung nützlich ist. Den entsprechenden Wert finden Sie in der Dokumentation zu Ihrem Spindelcontroller.
- *Drehzahl 1 und 2, PWM 1 und 2* - Die generierte Konfigurationsdatei verwendet eine einfache lineare Beziehung, um den PWM-Wert für einen bestimmten Drehzahlwert zu bestimmen. Wenn die Werte nicht bekannt sind, können sie bestimmt werden. Weitere Informationen finden Sie unter Festlegung der Spindle Kalibrierung.

Spindelsynchronisierte Bewegung

Wenn die entsprechenden Signale von einem Spindel-Encoder an LinuxCNC über HAL verbunden sind, unterstützt LinuxCNC Drehmaschine Gewindeschneiden. Diese Signale sind:

- *Spindle Index* – Ist ein Impuls, der einmal pro Spindelumdrehung auftritt.
- *Spindelphase A* - Dies ist ein Impuls, der an mehreren gleichmäßig beabstandeten Stellen auftritt, während sich die Spindel dreht.
- *Spindelphase B (optional)* - Dies ist ein zweiter Impuls, der auftritt, jedoch mit einem Versatz zur Spindelphase A. Die Vorteile der Verwendung von A und B sind Richtungserkennung, erhöhte Störfestigkeit und erhöhte Auflösung.

Wenn *Spindel Phase A* und *Spindel Index* auf dem Pinout erscheinen, sollten die folgenden Informationen eingegeben werden:

- *Use Spindle-At-Speed* - Mit Encoder-Feedback kann man wählen, ob LinuxCNC warten soll, bis die Spindel die befohlene Geschwindigkeit erreicht hat, bevor der Vorschub erfolgt. Wählen Sie diese Option und stellen Sie die *close enough* Skala ein.
- *Filterverstärkung der Drehzahlanzeige* - Einstellung zur Anpassung der Stabilität der visuellen Spindeldrehzahlanzeige.
- *Zyklen pro Umdrehung* - (enl. cycles per revolution) Die Anzahl der Zyklen des *Spindel A* Signals während einer Umdrehung der Spindel. Diese Option ist nur aktiviert, wenn ein Eingang auf *Spindel Phase A* gesetzt wurde
- *Maximale Drehzahl im Gewinde* - (engl. Maximum speed in thread) Die maximale Spindeldrehzahl, die beim Gewindeschneiden verwendet wird. Für eine hohe Spindeldrehzahl oder einen Spindel-Encoder mit hoher Auflösung ist ein niedriger Wert für *BASE_PERIOD* erforderlich.

Bestimmung der Spindelkalibrierung

Geben Sie die folgenden Werte auf der Seite Spindelkonfiguration ein:

Speed 1:	0	**PWM 1:**	0
Speed 2:	1000	**PWM 2:**	1

Beenden Sie die verbleibenden Schritte des Konfigurationsprozesses und starten Sie dann LinuxCNC mit Ihrer Konfiguration. Schalten Sie die Maschine ein und wählen Sie den Reiter MDI. Starten Sie die Spindeldrehung durch Eingabe von: *M3 S100*. Ändern Sie die Spindeldrehzahl, indem Sie eine andere S-Zahl eingeben: *S800*. Gültige Zahlen (zu diesem Zeitpunkt) reichen von 1 bis 1000.

Messen Sie für zwei verschiedene S-Zahlen die tatsächliche Spindeldrehzahl in U/min. Notieren Sie die S-Zahlen und die tatsächlichen Spindeldrehzahlen. Führen Sie StepConf erneut aus. Geben Sie für *Drehzahl* die gemessene Drehzahl und für *PWM* die S-Zahl geteilt durch 1000 ein.

Da die meisten Spindeltreiber in ihren Ansprechkurven etwas nichtlinear sind, ist es am besten, dies zu berücksichtigen:

- Stellen Sie sicher, dass die beiden Kalibrierungsdrehzahlen nicht zu nahe beieinander liegen.
- Vergewissern Sie sich, dass die beiden Kalibrierungsgeschwindigkeiten im Bereich der Geschwindigkeiten liegen, die Sie normalerweise beim Fräsen verwenden.

Wenn Ihre Spindel z. B. von 0 U/min bis 8000 U/min läuft, Sie aber in der Regel Drehzahlen zwischen 400 U/min (10 %) und 4000 U/min (100 %) verwenden, dann suchen Sie die PWM-Werte, die 1600 U/min (40 %) und 2800 U/min (70 %) ergeben.

3.1.8. Optionen

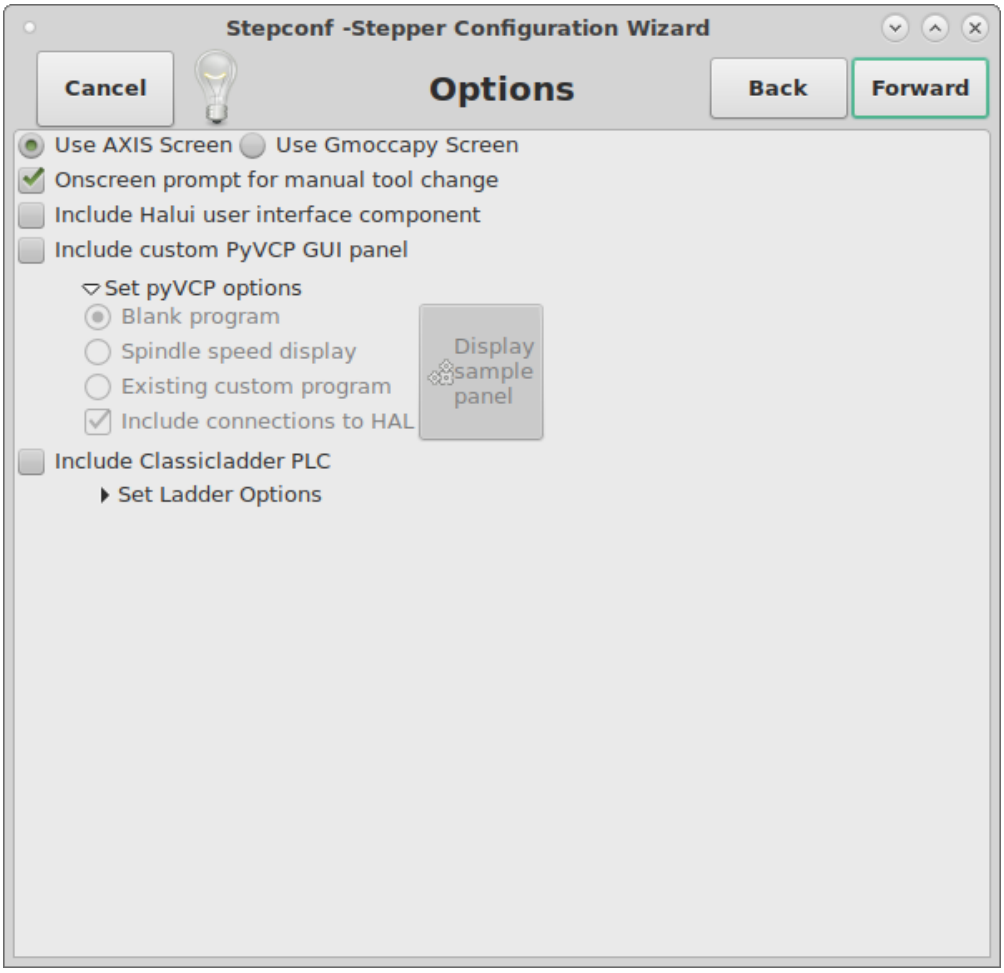


Figure 35. Erweiterte Optionen bei der Konfiguration

- *Include Halui* - Damit wird die Halui-Benutzerschnittstellenkomponente hinzugefügt. Siehe das HALUI Kapitel für weitere Informationen hierzu.
- *Include PyVCP* - Diese Option fügt die PyVCP-Panel-Basisdatei oder eine Beispieldatei zum Arbeiten hinzu. Siehe das PyVCP Kapitel für weitere Informationen.
- *Include ClassicLadder PLC* - Diese Option fügt die ClassicLadder PLC (Speicherprogrammierbare Steuerung) hinzu. Weitere Informationen finden Sie im Kapitel ClassicLadder.
- *Bildschirm-Aufforderung zu Werkzeugwechsel* - Wenn dieses Feld markiert ist, wird LinuxCNC Pause und fordern Sie auf, das Werkzeug zu wechseln, wenn *M6* angetroffen wird. Diese Funktion ist in der Regel nur sinnvoll, wenn Sie voreinstellbaren Werkzeugen haben.

3.1.9. Vollständige Maschinenkonfiguration

Klicken Sie auf *Übernehmen*, um die Konfigurationsdateien zu schreiben. Später können Sie dieses Programm erneut ausführen und die zuvor eingegebenen Einstellungen ändern.

3.1.10. Achsen Verfahrwege und Referenzpunkte

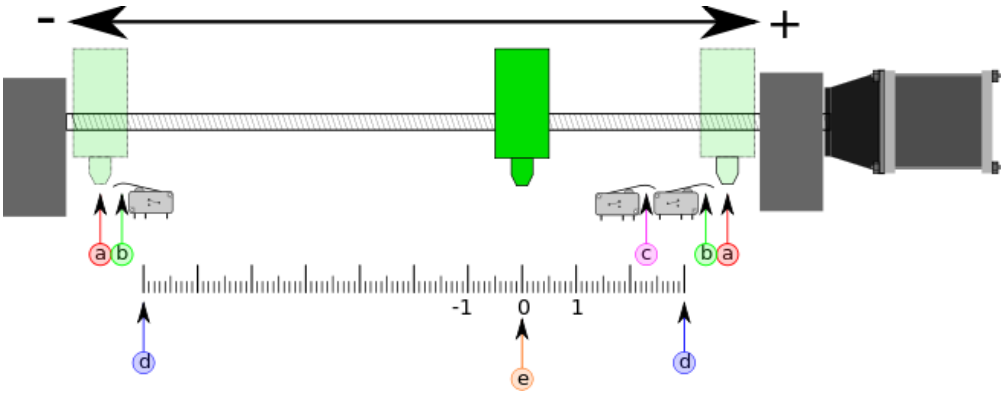


Figure 36. Achsen Verfahrwege und Referenzpunkt

Für jede Achse gibt es einen begrenzten Verfahrbereich. Das physische Ende des Verfahrwegs wird als Endanschlag (engl. hard stop) bezeichnet.

WARNING	[Bei Überschreitung eines mechanischen Festanschlags würde die Schnecke oder der Maschinenrahmen beschädigt werden!

Vor dem *hard stop* gibt es einen *Endschalter*. Wenn der Endschalter während des normalen Betriebs angetroffen wird, schaltet LinuxCNC den Motorverstärker ab. Der Abstand zwischen dem *harten Anschlag* und *Endschalter* muss lang genug sein, um einen unbestromten Motor zum Stillstand zu bringen.

Vor dem *Endschalter* gibt es ein *soft limit*. Dabei handelt es sich um eine Grenze, die nach der Referenzfahrt in der Software durchgesetzt wird. Wenn ein MDI-Befehl oder ein G-Code-Programm die weiche Grenze überschreiten würde, wird es nicht ausgeführt. Wenn eine manuelle Steuerung die Softgrenze überschreiten würde, wird diese an der Softgrenze beendet.

Der *Referenzschalter* (engl. home switch) kann an einer beliebigen Stelle innerhalb des Verfahrwegs (zwischen harten Anschlägen) platziert werden. Solange die externe Hardware die Motorverstärker nicht deaktiviert, wenn der Endschalter erreicht wird, kann einer der Endschalter als Referenzschalter verwendet werden.

Die *Nullposition* ist die Stelle auf der Achse, die im Maschinenkoordinatensystem 0 ist. Normalerweise liegt die *Nullposition* innerhalb der *weichen Grenzen*. Bei Drehmaschinen erfordert der Modus Konstante Schnittgeschwindigkeit, dass *X=0* dem Zentrum der Spindeldrehung entspricht, wenn keine Werkzeugkorrektur wirksam ist.

Die *Referenzpunktposition* (engl. home position) ist die Position innerhalb des Verfahrwegs, zu der die Achse am Ende der Referenzfahrt bewegt wird. Dieser Wert muss innerhalb der *weichen Grenzen* liegen. Insbesondere sollte die *Home Position* nie genau gleich einem *Soft Limit* sein.

Betrieb ohne Endschalter

Eine Maschine kann auch ohne Endschalter betrieben werden. In diesem Fall verhindern nur die Softlimits, dass die Maschine den Hardstop erreicht. Die Softlimits wirken erst, nachdem die Maschine referenziert worden ist.

Betrieb ohne Referenz-Endschalter

Eine Maschine kann auch ohne Referenzschalter betrieben werden. Wenn die Maschine Endschalter, aber keine Referenzschalter hat, ist es am besten, einen Endschalter als Referenzschalter zu verwenden (z.B. wählen Sie *Minimum Limit + Home X* in der Pinbelegung). Wenn die Maschine überhaupt keine Schalter hat oder die Endschalter aus einem anderen Grund nicht als Referenzschalter verwendet werden können, muss die Maschine nach Augenmaß oder mit Hilfe von Streichhölzern referenziert werden. Die Referenzfahrt nach Augenmaß ist nicht so wiederholbar wie die Referenzfahrt mit Schaltern, aber die Softlimits sind trotzdem nützlich.

Verdrahtungsoptionen für Referenz- und Endschalter

Die ideale Verdrahtung für externe Schalter wäre ein Eingang pro Schalter. Der PC-Parallelport bietet jedoch nur insgesamt 5 Eingänge, während es bei einer 3-Achsen-Maschine bis zu 9 Schalter gibt. Stattdessen werden mehrere Schalter auf verschiedene Weise miteinander verdrahtet, so dass eine geringere Anzahl von Eingängen erforderlich ist.

Die folgenden Abbildungen zeigen die allgemeine Idee der Verdrahtung mehrerer Schalter mit einem einzigen Eingangsstift. In jedem Fall, wenn ein Schalter betätigt wird, geht der Wert am INPUT von logisch HIGH auf LOW. LinuxCNC erwartet jedoch einen TRUE-Wert, wenn ein Schalter geschlossen ist, so dass die entsprechende *Invert*-Box auf der Pinout-Konfigurationsseite aktiviert werden muss. Der in den Diagrammen gezeigte Pull-Up-Widerstand zieht den Eingang auf HIGH, bis die Verbindung zur Masse hergestellt ist, dann geht der Eingang auf LOW. Andernfalls könnte der Eingang zwischen ein und aus schwanken, wenn der Schaltkreis offen ist. Für eine parallele Schnittstelle können Sie typischerweise 47 kΩ; verwenden.

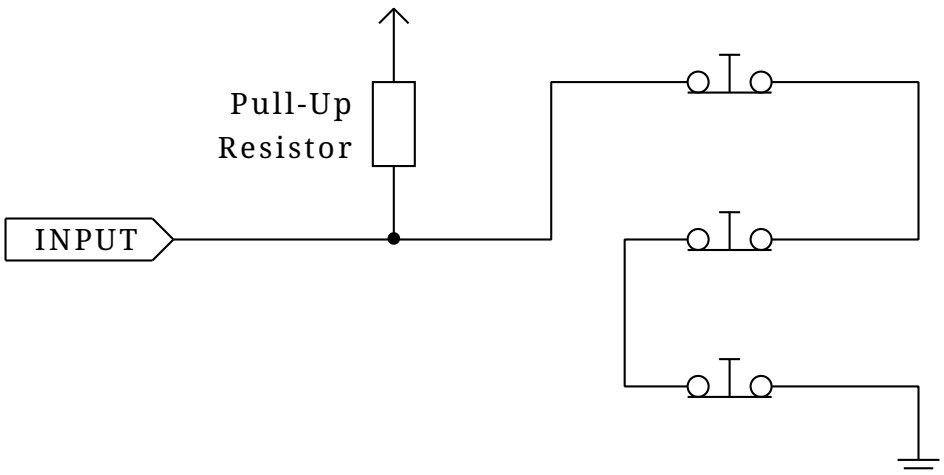


Figure 37. Normalerweise geschlossene Schalter (engl. normally closed, N/C) in Reihe verdrahtet (vereinfachtes Diagramm)

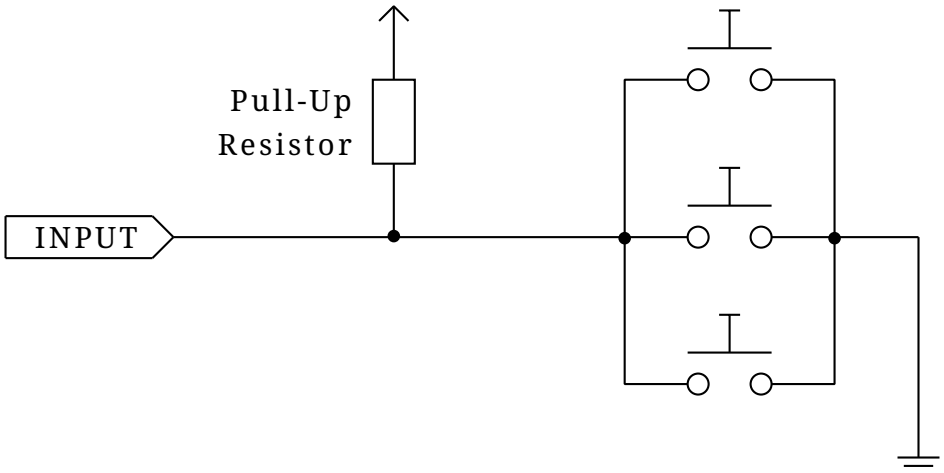


Figure 38. Normalerweise offene Schalter (engl. normally open switches, N/O) parallel verdrahtet (vereinfachte Darstellung)

Die folgenden Kombinationen von Schaltern sind in StepConf zulässig:

- Kombinierte Referenzschalter für alle Achsen
- Kombinierte Endschalter für alle Achsen
- Kombinieren beider Endschalter für eine Achse
- Kombinieren beider Endschalter und des Referenzschalters für eine Achse
- Kombinieren eines Endschalters und des Referenzschalters für eine Achse

Die letzten beiden Kombinationen sind auch geeignet, wenn der Typ Kontakt Referenz verwendet wird.

3.2. Mesa-Konfigurationsassistent

PnCconf wurde entwickelt, um Konfigurationen zu erstellen, die bestimmte Mesa *Anything I/O* Produkte verwenden.

Es kann Servo-Systeme mit geschlossenem Regelkreis oder Hardware-Schrittmachersysteme konfigurieren. Es verwendet einen ähnlichen *Assistenten* Ansatz wie StepConf (verwendet für Software-Stepping, parallel portgesteuerte Systeme).

PnCconf befindet sich noch im Entwicklungsstadium (Beta), daher gibt es noch einige Bugs und fehlende Funktionen. Bitte melden Sie Fehler und Vorschläge auf der LinuxCNC Forumsseite oder über die Mailing-Liste.

Bei der Verwendung von PnCconf gibt es zwei Denkansätze:

Wenn Sie sich entscheiden, Optionen zu ändern, laden Sie PnCconf neu und lassen Sie es die neuen Optionen konfigurieren. Dies funktioniert gut, wenn Ihr Rechner ziemlich standardmäßig ist und Sie benutzerdefinierte Dateien verwenden können, um nicht standardmäßige Funktionen hinzuzufügen. PnCconf versucht, Sie in dieser Hinsicht zu unterstützen.

Die andere Möglichkeit ist, PnCconf zu verwenden, um eine Konfiguration zu erstellen, die dem entspricht, was Sie wollen, und dann alles von Hand zu bearbeiten, um es an Ihre Bedürfnisse anzupassen. Dies wäre die Wahl, wenn Sie umfangreiche Änderungen über PnCconf's Umfang oder wollen einfach nur mit / lernen über LinuxCNC basteln müssen.

Mit den Schaltflächen "Vor", "Zurück" und "Abbrechen" können Sie durch die Seiten des Assistenten navigieren. Außerdem gibt es eine Hilfeschaltfläche, die einige Informationen zu den Seiten, Diagrammen und einer Ausgabeseite enthält.

TIP Die Hilfeseite von PnCconf sollte die aktuellsten Informationen und zusätzliche Details enthalten.

3.2.1. Schritt für Schritt Anleitung

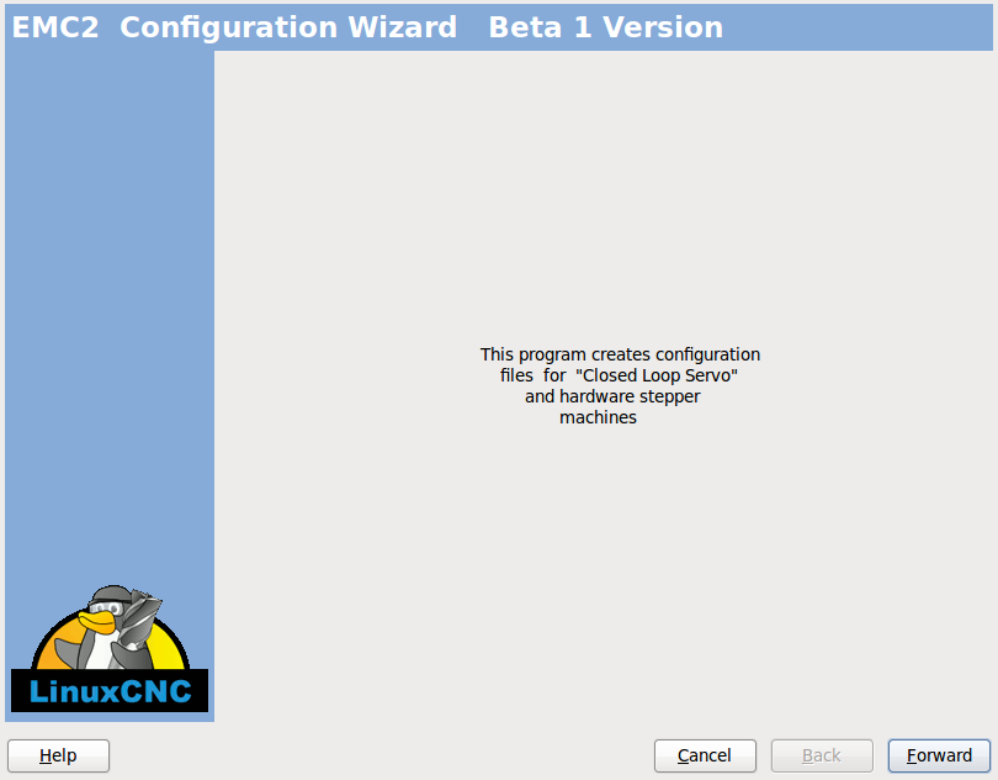


Figure 39. PnCconf Startfenster

3.2.2. Erstellen oder bearbeiten

Hier können Sie eine zuvor gespeicherte Konfiguration auswählen oder eine neue Konfiguration erstellen. Wenn Sie *Ändern einer Konfiguration* wählen und dann *Weiter* drücken, wird eine Dateiauswahlbox angezeigt. PnCconf wählt Ihre zuletzt gespeicherte Datei vor. Wählen Sie die Konfiguration, die Sie bearbeiten möchten. Wenn Sie Änderungen an der Haupt-HAL oder den INI-Dateien vorgenommen haben, überschreibt **PnCconf** diese Dateien und die Änderungen gehen verloren. Einige Dateien werden nicht überschrieben und PnCconf fügt einen Hinweis in diese Dateien ein. Außerdem können Sie Desktop-Verknüpfungen und Startoptionen auswählen. Eine Desktop-Verknüpfung legt ein Ordnersymbol auf dem Desktop ab, das auf Ihre neuen Konfigurationsdateien verweist. Andernfalls müssten Sie in Ihrem Home-Ordner unter linuxcnc/configs suchen.

Ein Desktop Launcher fügt ein Symbol auf dem Desktop hinzu, um Ihre Konfiguration direkt zu starten. Sie können sie auch vom Hauptmenü aus starten, indem Sie den Configuration Selector *LinuxCNC* im CNC-Menü verwenden und den Namen Ihrer Konfiguration auswählen.

3.2.3. Grundlegende Informationen zur Maschine

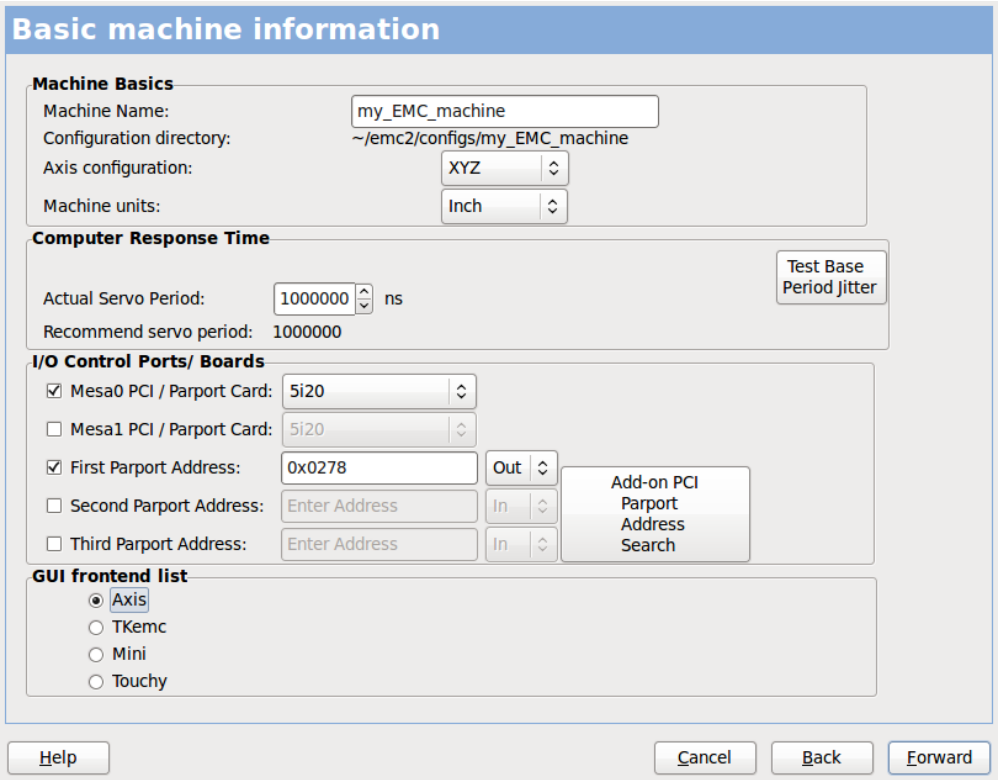


Figure 40. PnCconf Basic

Maschinengrundlagen

Wenn Sie einen Namen mit Leerzeichen verwenden, ersetzt PnCconf die Leerzeichen durch Unterstriche (als lockere Regel gilt, dass Linux keine Leerzeichen in Namen mag). Durch Wahl einer Achsenkonfiguration bestimmen Sie, welche Art von Maschine Sie bauen und welche Achsen verfügbar sind. Die Auswahl der "Maschineneinheiten" (engl. machine units) ermöglicht die Eingabe von metrischen oder imperialen Einheiten im weiteren Verlauf der Konfiguration.

TIP Standardwerte werden bei der Verwendung in metrisch nicht konvertiert, stellen Sie also sicher, dass es sich um vernünftige Werte handelt!

Reaktionszeit des Computers

The servo period sets the heart beat of the system. Latency describes the difference between the time that the system is scheduled to perform and action and the time that it actually does perform the action. Just like a railroad, LinuxCNC requires everything on a very tight and consistent timeline or bad things happen. LinuxCNC requires and uses a *real-time* operating system, which just means it has a low-latency (lateness) response time. When LinuxCNC requires and is performing calculations, it cannot be interrupted by lower priority requests (such as user input to screen buttons or drawing etc).

Testing the latency is crucial and a key thing to check before proceeding further. Please follow the

directions on the Latency Test page before proceeding further.

Jetzt sind wir mit der Latenzzeit zufrieden und müssen eine Servoperiode wählen. In den meisten Fällen ein Servo-Periode von 1000000 ns ist in Ordnung (das gibt eine 1 kHz Servo Berechnungsrate - 1000 Berechnungen pro Sekunde). Wenn Sie den Aufbau eines geschlossenen Regelkreises Servo-System, das Drehmoment (Strom) steuert, anstatt Geschwindigkeit (Spannung), so wäre eine schnellere Rate besser - etwa 200000 (5 kHz Berechnungsrate). Das Problem mit der Senkung der Servo-Rate ist, dass weniger Zeit für den Computer zur Verfügung steht, um andere Dinge neben LinuxCNC zu berechnen. Typischerweise wird die Anzeige (GUI) weniger reaktionsschnell. Sie müssen sich für ein Gleichgewicht entscheiden. Denken Sie daran, dass, wenn Sie Ihre Closed-Loop-Servo-System abstimmen (engl. tune) und dann die Servo-Periode ändern, Sie anschließend voraussichtlich wieder Ihr Servo-System abstimmen müssen.

E/A-Steueranschlüsse/Karten (engl. I/O Control Ports/Boards)

PnCconf ist in der Lage, Maschinen mit bis zu zwei Mesa-Karten und drei parallelen Anschlüssen zu konfigurieren. Parallele Schnittstellen können nur für einfache E/A mit niedriger Geschwindigkeit (Servorate) verwendet werden.

Mesa

You must choose at least one Mesa board as PnCconf will not configure the parallel ports to count encoders or output step or PWM signals. The mesa cards available in the selection box are based on what PnCconf finds for firmware on the systems. PnCconf does not load firmware onto the card - it only reads the firmware already present to learn which pins and components exist, and writes the HAL/INI files to match. The card must already run the firmware you intend to configure. Firmware is loaded beforehand with the `mesaflash` utility, e.g. `mesaflash --device 7i92 --write 7i92t_xxx.bit`; the card must be power-cycled after flashing. Selecting a firmware in PnCconf that differs from the one actually flashed results in HAL load errors or wrong pin names. There are options to add custom firmware and/or *blacklist* (ignore) some firmware or boards using a preference file. If no firmware is found PnCconf will show a warning and use internal sample firmware - no testing will be possible. One point to note is that if you choose two PCI Mesa cards there currently is no way to predict which card is 0 and which is 1 - you must test - moving the cards could change their order. If you configure with two cards both cards must be installed for tests to function.

Parallelport

Bis zu 3 parallele Anschlüsse (auch parallele Schnittstelle oder engl. Kurzform parports) können als einfache E/A verwendet werden. Sie müssen die Adresse des Parports festlegen. Sie können entweder das Linux-Nummerierungssystem für parallele Anschlüsse (0, 1 oder 2) oder die tatsächliche Adresse eingeben. Die Adresse für einen On-Board-Parport ist oft 0x0278 oder 0x0378 (in Hexadezimal geschrieben), kann aber auf bei denr BIOS-Einstellungen gefunden werden. Die BIOS-Seite finden Sie, wenn Sie Ihren Computer zum ersten Mal starten. Sie müssen eine Taste drücken, um sie zu öffnen (z. B. F2). Auf der BIOS-Seite finden Sie die Adresse des parallelen Anschlusses und können den Modus einstellen, z. B. SPP, EPP usw. Bei einigen Computern wird diese Information während des Starts einige Sekunden lang angezeigt. Bei PCI-Parallelport-Karten kann die Adresse durch Drücken der Schaltfläche *parport address search* ermittelt werden. Daraufhin wird die Hilfe-Ausgabeseite mit einer Liste aller PCI-Geräte angezeigt, die gefunden werden können. Darin sollte ein Verweis auf ein Parallelport-Gerät mit einer Liste von Adressen enthalten sein. Eine dieser Adressen sollte funktionieren. Nicht alle PCI-Parallelports funktionieren ordnungsgemäß. Beide Typen können als *in*

(maximale Anzahl von Eingangspins) oder *out* (maximale Anzahl von Ausgangspins) ausgewählt werden.

GUI-Frontend-Liste

Dies legt die grafischen Bildschirme fest, die LinuxCNC verwenden wird. Jeder von ihnen hat verschiedene Optionen.

ACHSE

- Vollständig unterstützt Drehmaschinen.
- ist das am weitesten entwickelte und am häufigsten verwendete Frontend
- ist für die Verwendung mit Maus und Tastatur konzipiert
- basiert auf tkinter und integriert daher unkompliziert mit PyVCP (Python-basierte virtuelle Kontrollfelder).
- hat ein grafisches 3D-Fenster.
- ermöglicht die Integration von VCP auf der Seiten- oder in der mittleren Registerkarte

TkLinuxCNC

- kontrastreicher hellblauer Bildschirm
- separates Grafikfenster
- keine VCP-Integration

Touchy

- Touchy wurde für die Verwendung mit einem Touchscreen, einigen minimalen physischen Schaltern und einem MPG-Rad konzipiert.
- erfordert Zyklus-Start-, Abbruch- und Einzelschritt-Signale und -Tasten
- Außerdem muss das Handrad-Jogging mit gemeinsamer Achse ausgewählt werden.
- ist GTK-basiert und integriert daher GladeVCP (virtuelle Kontrollfelder) auf unkompiziere Weise.
- ermöglicht die Integration von VCP-Panels in die mittlere Registerkarte
- hat kein grafisches Fenster
- Das Aussehen kann mit benutzerdefinierten Designs geändert werden

QtPlasmaC

- voll funktionsfähige Plasmac-Konfiguration auf der Grundlage der QtVCP-Infrastruktur.
- Maus-/Tastaturbedienung oder Touchscreen-Bedienung
- keine VCP-Integration

3.2.4. Externe Konfiguration

Auf dieser Seite können Sie externe Steuerungen auswählen, z. B. für Jogging oder Overrides.

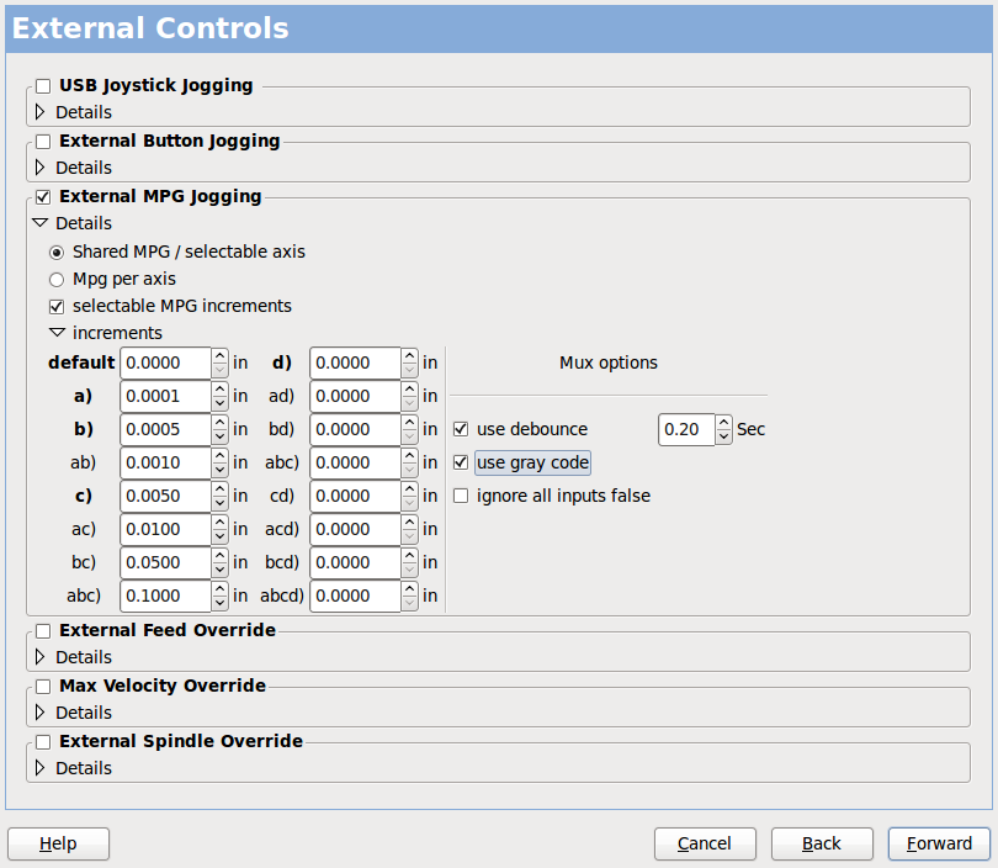


Figure 41. Externe Steuerelemente

Wenn Sie einen Joystick für Jogging wählen, so muß dieser immer eingesteckt sein, um LinuxCNC zu nutzen. Um die analogen Sticks für Jogging zu nutzen, werden Sie wahrscheinlich einigen HAL-Code selber hinzufügen müssen. Handrad (auch Impulsgenerator oder engl. MPG)-Jogging erfordert einen Impulsgeber, der mit einem MESA-Geberzähler verbunden ist. Override-Steuerungen können entweder einen Impulsgenerator oder einem Schalter (z. B. einen Drehschalter) verwenden. Externe Tasten können mit einem schalterbasierten OEM-Joystick verwendet werden.

Joystick-Joggen

Hierzu muss eine benutzerdefinierte *device rule* im System installiert werden. Dies ist eine Datei, die LinuxCNC verwendet, um eine Verbindung zu Linux's Geräte (engl. device)-Liste herstellt. PnCconf wird helfen, diese Datei zu anzulegen.

- *Suche nach Geräteregeln* durchsucht das System nach Regeln. Sie können diese Funktion verwenden, um den Namen von Geräten zu finden, die Sie bereits mit PnCconf erstellt haben.
- Mit *Add a device rule* können Sie ein neues Gerät konfigurieren, indem Sie den Aufforderungen folgen. Sie müssen Ihr Gerät zur Verfügung haben.
- Mit *test device* können Sie ein Gerät laden, dessen Pin-Namen sehen und seine Funktionen mit halmeter überprüfen.

Joystick-Jogging verwendet die Komponenten HALUI und hal_input.

Externe Tasten

Ermöglicht das Joggen der Achse mit einfachen Tasten mit einer bestimmten Geschwindigkeit. Wahrscheinlich am besten für schnelles Joggen geeignet.

Handrad (engl. MPG) Jogging

Ermöglicht die Verwendung eines Handrads (manuellen Impulsgebers, MPG), um die Achsen der Maschine zu bewegen.

Handräder (MPGs) sind häufig in kommerziellen Maschinen zu finden. Sie geben Quadraturimpulse aus, die mit einem MESA-Encoder-Zähler gezählt werden können. PnCconf ermöglicht ein Rad pro Achse oder eines gemeinsam für alle Achsen. Es ermöglicht die Auswahl von Jog-Geschwindigkeiten über Schalter oder eine voreingestellte feste Geschwindigkeit.

Für die Option der wählbaren Inkremente wird die Komponente mux16 verwendet. Diese Komponente verfügt über Optionen wie Entprellung und Gray-Code, um den rohen Schaltereingang zu filtern.

Neufestlegungen (engl. overrides)

PnCconf ermöglicht die Neufestsetzung von Vorschubgeschwindigkeiten und/oder Spindeldrehzahlen über ein Handrad (MPG) oder Schalter (z. B. Drehschalter).

3.2.5. GUI-Konfiguration

Hier können Sie die Standardeinstellungen für die Bildschirme, fügen Sie virtuelle Bedienfelder (VCP), und stellen Sie einige LinuxCNC Optionen.

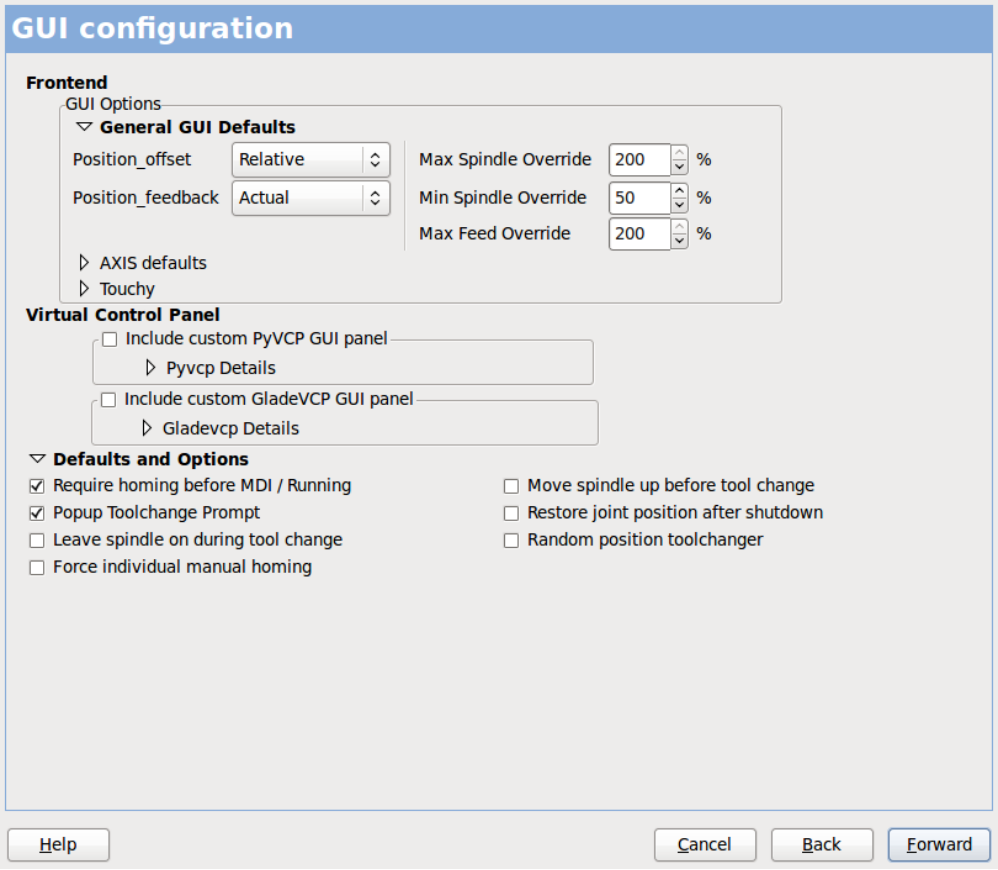


Figure 42. GUI-Konfiguration

Front-End GUI-Optionen

Die Standardoptionen ermöglichen die Auswahl allgemeiner Standardeinstellungen für jeden Anzeigebildschirm.

AXIS-Standardwerte sind AXIS-spezifische Optionen. Wenn Sie die Optionen Größe, Position oder Maximieren erzwingen wählen, fragt PnCconf, ob eine Voreinstellungsdatei (.axisrc) überschrieben werden darf. Sofern Sie nicht manuell Befehle zu dieser Datei hinzugefügt haben, ist es in Ordnung, dies zuzulassen. Position und force max können verwendet werden, um AXIS auf einen zweiten Monitor zu verschieben, wenn das System dazu in der Lage ist.

Touchy-Standardeinstellungen sind Optionen, die spezifisch für Touchy sind. Die meisten Optionen von Touchy'können während der Ausführung von Touchy über die Einstellungsseite geändert werden. Touchy verwendet GTK, um seinen Bildschirm zu zeichnen, und GTK unterstützt Themen. Themes steuern das grundlegende Aussehen und die Bedienung eines Programms. Sie können Themes aus dem Netz herunterladen oder sie selbst bearbeiten. Es gibt eine Liste der aktuellen Themen auf dem Computer, aus der Sie auswählen können. Damit ein Teil des Textes besser zur Geltung kommt, können Sie in PnCconf die Standardeinstellungen der Themes's überschreiben. Die Optionen position und force max können verwendet werden, um Touchy auf einen zweiten Monitor zu verschieben, wenn das System dazu in der Lage ist.

QtPlasmaC-Optionen sind spezifisch für QtPlasmac, alle allgemeinen Optionen, die nicht benötigt werden, sind deaktiviert. Wenn QtPlasmac ausgewählt wird, ist der folgende Bildschirm ein Bildschirm zur Einstellung der Benutzertasten, der spezifisch für QtPlasmaC ist, und die VCP-Optionen sind nicht verfügbar.

VCP Optionen

Virtuelle Bedienfelder ermöglichen das Hinzufügen von benutzerdefinierten Bedienelementen und Anzeigen auf dem Bildschirm. AXIS und Touchy können diese Steuerelemente an bestimmten Positionen in den Bildschirm integrieren. Es gibt zwei Arten von VCPs - PyVCP, das *Tkinter* zum Zeichnen des Bildschirms verwendet und GladeVCP, das *GTK* zum Zeichnen des Bildschirms verwendet.

PyVCP

PyVCPs Bildschirm XML-Datei kann nur von Hand erstellt werden. PyVCPs passen natürlich mit AXIS, da sie beide TKinter verwenden.

Es werden HAL-Pins erstellt, an die sich der Benutzer in seiner eigenen HAL-Datei anschließen kann. Es gibt ein Beispiel für ein Spindeldisplay, das der Benutzer unverändert verwenden oder darauf aufbauen kann. Sie können eine leere Datei auswählen, zu der Sie später Ihre Steuerelemente *Widgets* hinzufügen können, oder Sie wählen ein Beispiel für eine Spindelanzeige zur Anzahl der Spindeldrehzahl und zur Anzeige, ob die Spindel die gewünschte Drehzahl erreicht hat.

PnCconf verbindet die richtigen HAL-Pins der Spindelanzeige für Sie. Wenn Sie AXIS verwenden, wird das Panel auf der rechten Seite integriert. Wenn Sie AXIS nicht verwenden, wird das Panel separat *stand-alone* vom Front-End-Bildschirm angezeigt.

Sie können die Geometrieoptionen verwenden, um das Panel zu vergrößern und zu verschieben, z. B. um es auf einen zweiten Bildschirm zu verschieben, wenn das System dazu in der Lage ist. Wenn Sie auf die Schaltfläche *Mustertafel anzeigen* klicken, werden die Größen- und Platzierungsoptionen beachtet.

GladeVCP

GladeVCPs passen natürlich in den Touchy-Bildschirm, da beide GTK verwenden, um sie zu zeichnen,

aber durch die Änderung von GladeVCPs Thema kann es möglich werden, dies optisch ziemlich gut in AXIS zu integrieren (versuchen Sie Redmond).

Es verwendet einen grafischen Editor, um seine XML-Dateien zu erstellen. Es werden HAL-Pins erstellt, mit denen sich der Benutzer innerhalb seiner eigenen HAL-Datei verbinden kann.

GladeVCP ermöglicht auch eine viel anspruchsvollere (und kompliziertere) Programmierinteraktion, die PnCconf derzeit nicht nutzt (siehe GLADE VCP im Handbuch).

PnCconf verfügt über Beispielpanels, die der Benutzer unverändert verwenden oder auf denen er aufbauen kann. Mit GladeVCP können Sie in PnCconf verschiedene Optionen für Ihre Beispielanzeige auswählen.

Wählen Sie unter "Beispieloptionen" die gewünschten Optionen aus. Die Nulltasten verwenden HALUI-Befehle, die Sie später im HALUI-Abschnitt bearbeiten können.

Auto Z Touch-Off erfordert auch das klassische Leiter-Touch-Off-Programm und einen ausgewählten Sondeneingang. Es erfordert eine leitfähige Touch-Off-Platte und ein geerdetes leitfähiges Werkzeug. Eine Idee, wie es funktioniert, finden Sie unter:

https://wiki.linuxcnc.org/cgi-bin/wiki.pl?ClassicLadderExamples#Single_button_probe_touchoff

Unter „Anzeigeoptionen“ können Größe, Position und max. Kraft auf einem „eigenständigen“ Panel verwendet werden, um beispielsweise den Bildschirm auf einem zweiten Monitor zu platzieren, wenn das System dazu in der Lage ist.

Sie können ein GTK-Thema auswählen, welches das grundlegende Erscheinungsbild des Panels festlegt. Normalerweise möchten Sie, dass dies mit dem Front-End-Bildschirm übereinstimmt. Diese Optionen werden verwendet, wenn Sie auf die Schaltfläche "Beispiel anzeigen" klicken. Mit GladeVCP können Sie je nach Front-End-Bildschirm auswählen, wo das Panel angezeigt werden soll.

Bei AXIS kann er in der Mitte oder auf der rechten Seite stehen, bei Touchy in der Mitte.

Standardeinstellungen und Optionen

- Referenzfahrt vor Manueller Dateneingabe / Ausführung erforderlich machen
  - Wenn Sie möchten, dass die Maschine vor der Referenzfahrt bewegt werden kann, deaktivieren Sie dieses Kontrollkästchen.
- Popup-Tool-Eingabeaufforderung
  - Wählen Sie zwischen einer Bildschirmabfrage für Werkzeugwechsel oder dem Export von Standardsignalnamen für eine benutzerdefinierte Werkzeugwechsler-HAL-Datei
- Lassen Sie die Spindel während des Werkzeugwechsels eingeschaltet:
  - Verwendet für Drehbänke
- Individuelle manuelle Referenzfahrt erzwingen
- Spindel vor dem Werkzeugwechsel nach oben fahren
- Wiederherstellung der Gelenkposition nach Abschaltung
  - Verwendet für nichttriviale kinematische Maschinen

- Werkzeugwechsler mit zufälliger (engl. random) Position
  - Wird für Werkzeugwechsler verwendet, die das Werkzeug nicht in dieselbe Tasche zurückbringen. Um Werkzeugwechsler zu unterstützen, müssen Sie einen eigenen HAL-Code hinzufügen.

3.2.6. Mesa-Konfiguration

Die Mesa-Konfigurationsseiten erlauben es, verschiedene Firmwares zu verwenden. Auf der Basisseite haben Sie eine Mesa-Karte ausgewählt. Hier wählen Sie die verfügbare Firmware aus und bestimmen, welche und wie viele Komponenten verfügbar sind.

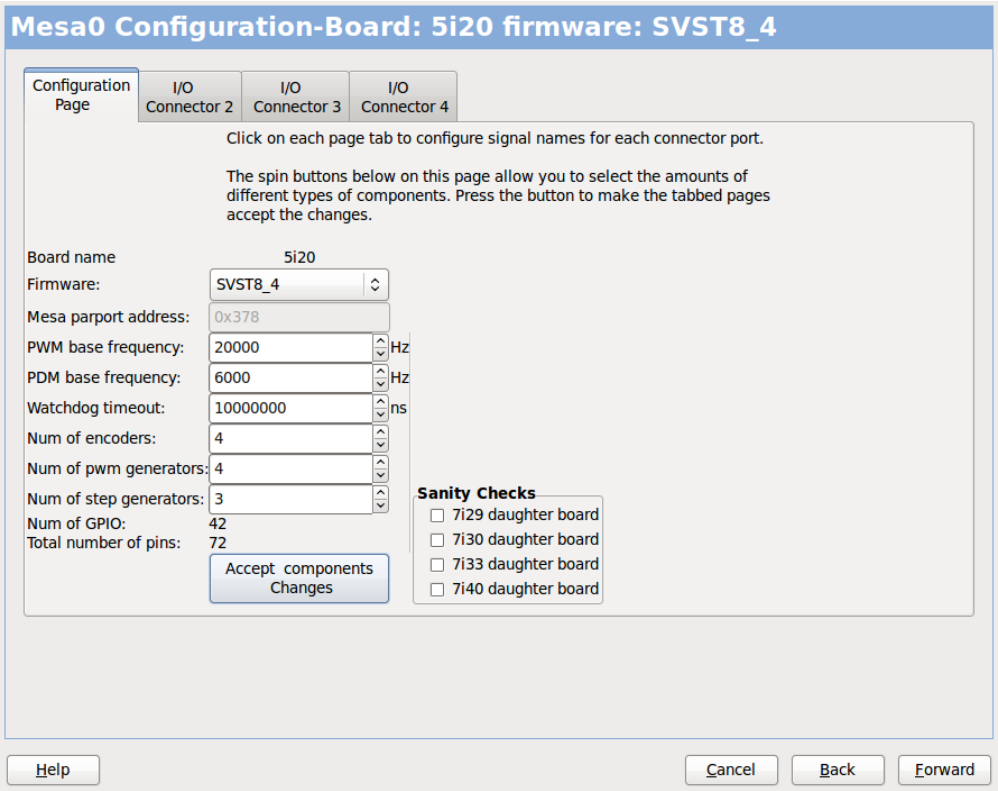


Figure 43. Mesa Board Konfiguration

Die Parport-Adresse wird nur mit der Mesa-Parport-Karte, der 7i43, verwendet. Ein On-Board-Parallelport verwendet normalerweise 0x278 oder 0x378, obwohl Sie in der Lage sein sollten, die Adresse auf der BIOS-Seite zu finden. Bei der 7i43 muss die parallele Schnittstelle den EPP-Modus verwenden, der wiederum im BIOS eingestellt wird. Bei Verwendung einer PCI-Parallelschnittstelle kann die Adresse über die Suchfunktion auf der Basisseite gesucht werden.

NOTE Viele PCI-Karten unterstützen das EPP-Protokoll nicht richtig.

Die PDM-PWM- und 3PWM-Basisfrequenz bestimmt das Gleichgewicht zwischen Restwelligkeit und Linearität. Bei der Verwendung von Mesa-Tochterkarten sollten die Unterlagen für die Karte Empfehlungen enthalten.

IMPORTANT Es ist wichtig, diese zu beachten, um Schäden zu vermeiden und die beste Leistung zu erzielen.

```
Der 7i33 benötigt PDM und eine PDM-Basisfrequenz von 6 MHz
Der 7i29 benötigt PWM und eine PWM-Basisfrequenz von 20 kHz
Das 7i30 erfordert PWM und eine PWM-Basisfrequenz von 20 kHz
Das 7i40 erfordert PWM und eine PWM-Basisfrequenz von 50 kHz
Das 7i48 benötigt UDM und eine PWM-Basisfrequenz von 24 kHz
```

Watchdog-Zeitüberschreitung

wird verwendet, um einzustellen, wie lange das MESA-Board wartet, bevor es die Ausgänge abschaltet, wenn die Kommunikation mit dem Computer unterbrochen wird. Bitte denken Sie daran, dass Mesa *active low* Ausgänge verwendet, d.h. wenn der Ausgangspin eingeschaltet wurde, dann ist er niedrig (ca. 0 Volt) und wenn er ausgeschaltet ist, dann ist der Ausgang hoch (ca. 5 Volt), stellen Sie sicher, dass Ihre Ausrüstung im ausgeschalteten Zustand (z.B. wenn der Watchdog einmal beißt) sicher ist.

Anzahl der Codierer/PWM-Generatoren/STEP-Generatoren

Sie können die Anzahl der verfügbaren Komponenten auswählen, indem Sie nicht verwendete Komponenten abwählen. Nicht alle Komponententypen sind mit jeder Firmware verfügbar.

Wenn Sie weniger als die maximale Anzahl von Komponenten wählen, können Sie mehr GPIO-Pins erhalten. Bei der Verwendung von Tochterkarten sollten Sie darauf achten, dass Sie keine Pins abwählen, die von der Karte verwendet werden. Zum Beispiel unterstützt manche Firmware zwei 7i33-Karten. Wenn Sie nur eine haben, können Sie genügend Komponenten abwählen, um den Anschluss zu nutzen, der die zweite 7i33 unterstützt. Die Komponenten werden numerisch mit der höchsten Nummer zuerst abgewählt, dann abwärts, ohne eine Nummer zu überspringen. Wenn die Komponenten dadurch nicht dort sind, wo Sie sie haben wollen, müssen Sie eine andere Firmware verwenden. Die Firmware diktiert, wo, was und die maximale Menge der Komponenten. Benutzerdefinierte Firmware ist möglich, fragen Sie freundlich, wenn Sie die LinuxCNC Entwickler und Mesa kontaktieren. Die Verwendung benutzerdefinierter Firmware in PnCconf erfordert spezielle Verfahren und ist nicht immer möglich - obwohl ich versuche, PnCconf so flexibel wie möglich zu machen.

Nachdem Sie all diese Optionen ausgewählt haben, drücken Sie die Schaltfläche *Accept Component Changes* und PnCconf aktualisiert die E/A-Setup-Seiten. Abhängig von der Mesa-Karte werden nur E/A-Registerkarten für verfügbare Anschlüsse angezeigt.

3.2.7. Mesa I/O-Einrichtung

Die Registerkarten werden zur Konfiguration der Eingangs- und Ausgangspins der Mesa-Karten verwendet. Mit PnCconf können Sie benutzerdefinierte Signalnamen zur Verwendung in benutzerdefinierten HAL-Dateien erstellen.

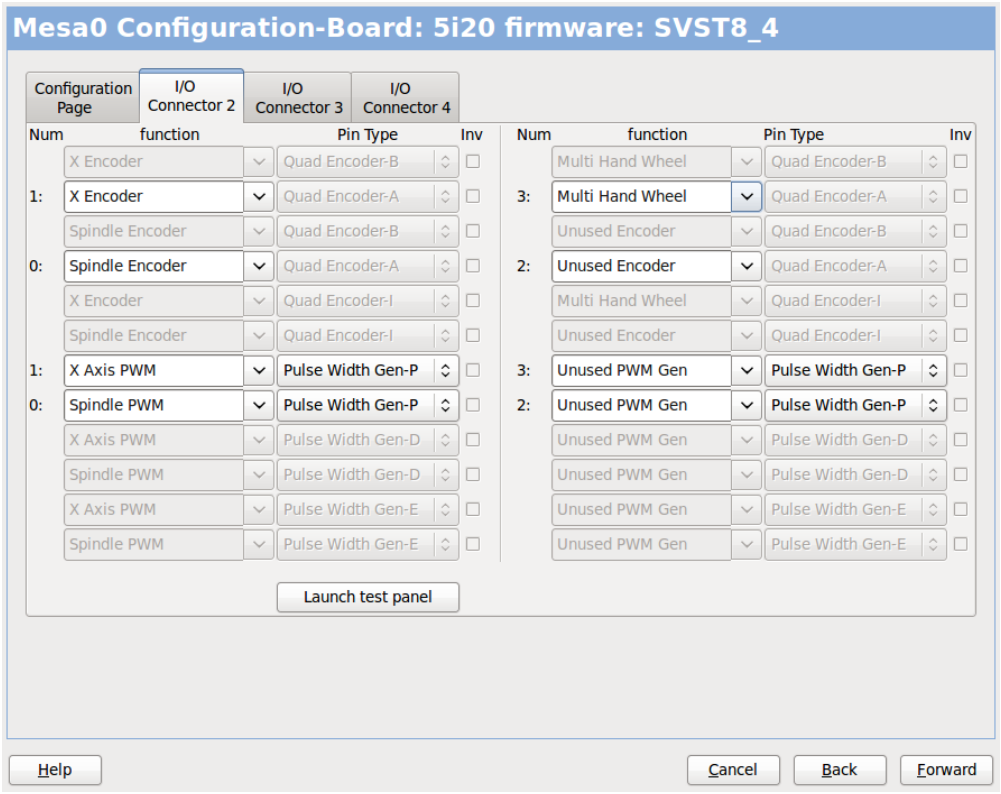


Figure 44. Mesa I/O C2 Einrichtung

Auf dieser Registerkarte mit dieser Firmware sind die Komponenten für eine 7i33 Tochterplatine eingestellt, die normalerweise mit Servos mit geschlossenem Regelkreis verwendet wird. Beachten Sie, dass die Komponentennummern der Encoderzähler und PWM-Treiber nicht in numerischer Reihenfolge sind. Dies entspricht den Anforderungen für die Tochterkarte.

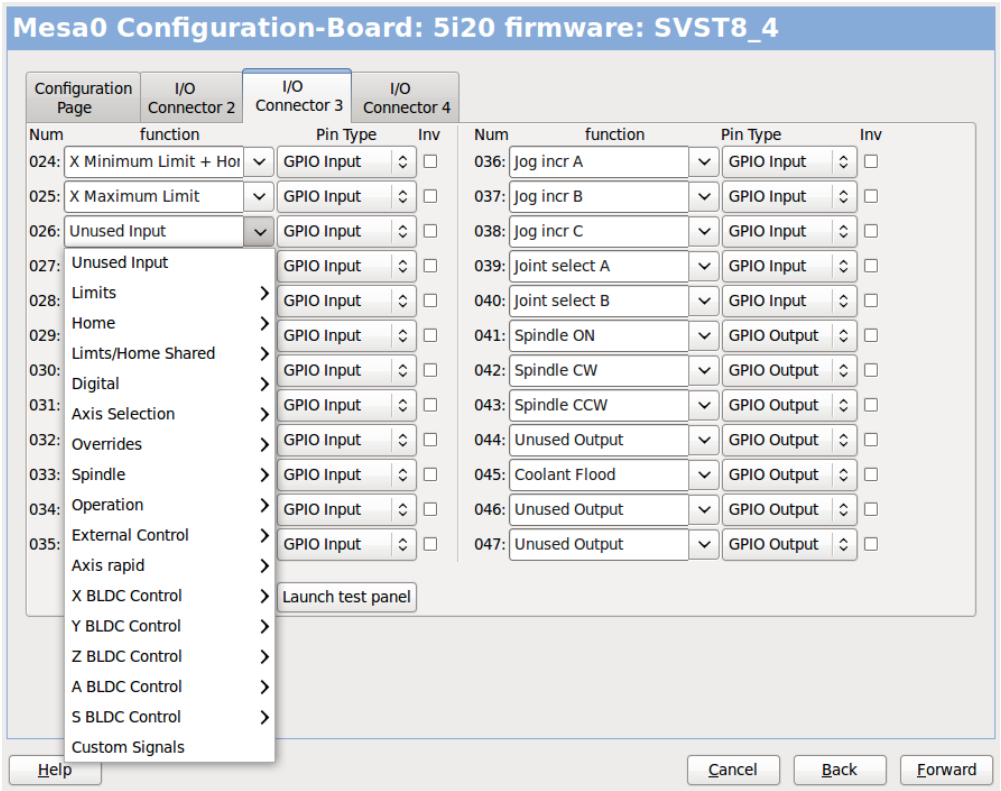


Figure 45. Mesa I/O C3 Einrichtung

Auf dieser Registerkarte sind alle Pins GPIO. Beachten Sie die 3-stelligen Nummern - sie entsprechen der HAL-Pin-Nummer. GPIO-Pins können als Eingang oder Ausgang gewählt werden und können invertiert werden.

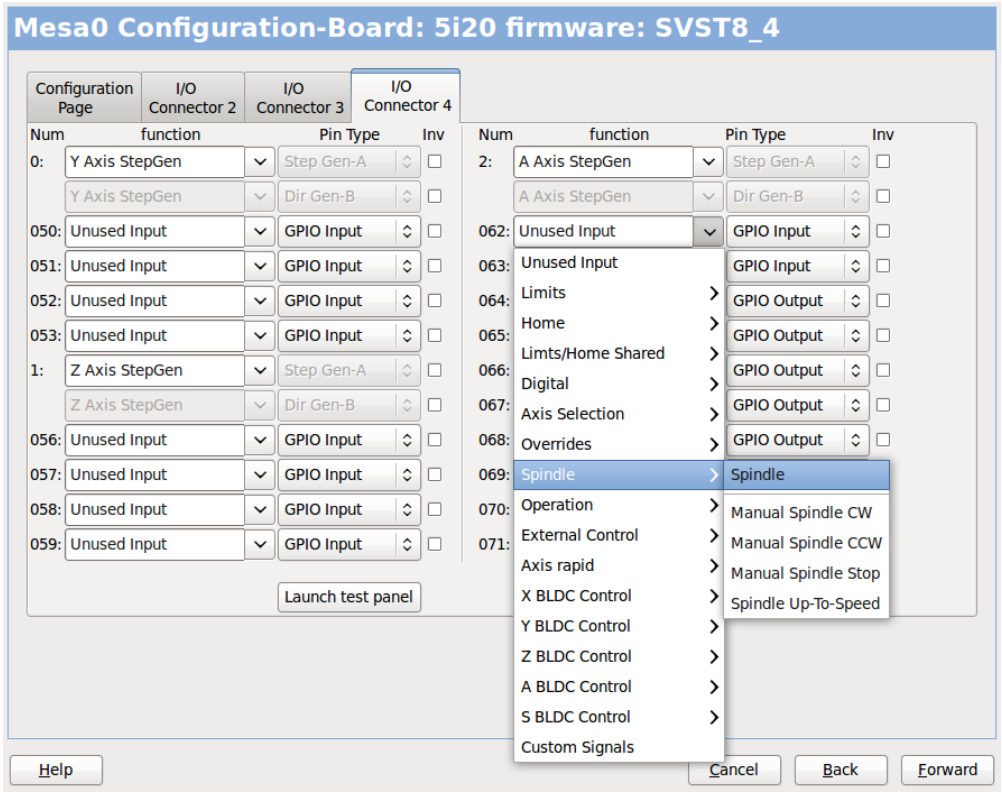
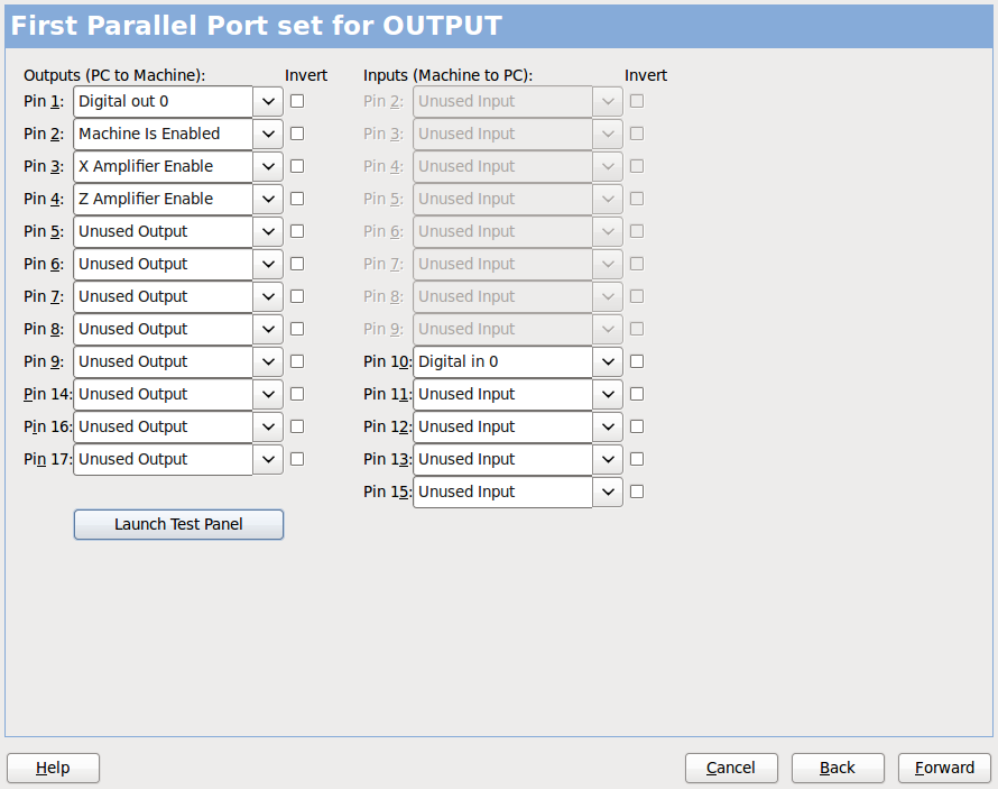


Figure 46. Mesa I/O C4 Einrichtung

Auf dieser Registerkarte gibt es eine Mischung aus Schrittgeneratoren und GPIO. Die Ausgangs- und Richtungspins der Schrittgeneratoren können invertiert werden. Beachten Sie, dass die Invertierung eines Step-Gen-A-Pins (des Step-Ausgangspins) das Step-Timing verändert. Es sollte dem entsprechen, was Ihr Controller erwartet.

3.2.8. Konfiguration des parallelen Anschlusses



Der Parallelport kann für einfache E/A verwendet werden, ähnlich wie die GPIO-Pins von Mesa's.

3.2.9. Konfiguration der Achsen

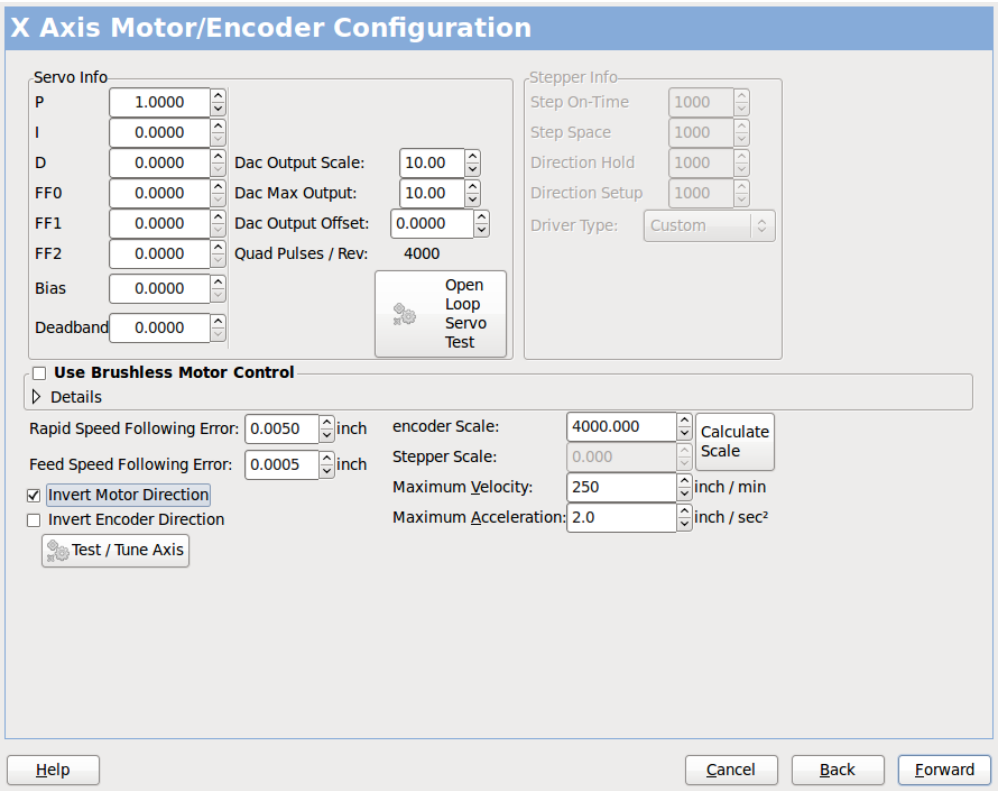


Figure 47. Konfiguration des Achsantriebs

Diese Seite ermöglicht das Konfigurieren und Testen der Motor- und/oder Encoderkombination. Bei Verwendung eines Servomotors ist ein Open-Loop-Test verfügbar, bei Verwendung eines Schrittmotors

ein Tuning-Test.

Open-Loop-Test

Ein Open-Loop-Test ist wichtig, da er die Richtung von Motor und Encoder bestätigt. Der Motor sollte die Achse in die positive Richtung bewegen, wenn die positive Taste gedrückt wird, und auch der Encoder sollte in die positive Richtung zählen. Die Achsenbewegung sollte den Fußnoten des Machinery's Handbook folgen:["Achsennomenklatur" im Kapitel "Numerische Steuerung" im "Machinery's Handbook", herausgegeben von Industrial Press,] sonst macht die grafische AXIS-Anzeige nicht viel Sinn. Hoffentlich können die Hilfeseite und die Diagramme helfen, dies herauszufinden. Beachten Sie, dass die Richtungen der Achsen auf der Bewegung des Werkzeugs und nicht auf der Bewegung des Tisches basieren. Beim Open-Loop-Test gibt es keine Beschleunigungsrampe, beginnen Sie also mit niedrigeren DAC-Zahlen. Durch Bewegen der Achse über eine bekannte Strecke kann man die Skalierung des Encoders bestätigen. Der Encoder sollte auch ohne aktivierten Verstärker zählen, je nachdem wie der Encoder mit Strom versorgt wird.

WARNING	Wenn Motor und Encoder die Zählrichtung nicht übereinstimmen, läuft das Servo bei PID-Regelung weg.

Da die PID-Einstellungen derzeit nicht in PnCconf getestet werden können, sind die Einstellungen eigentlich für die erneute Bearbeitung einer Konfiguration gedacht - geben Sie Ihre getesteten PID-Einstellungen ein.

DAC-Skala

DAC-Skalierung, maximaler Ausgang und Offset werden verwendet, um den DAC-Ausgang anzupassen.

DAC berechnen

Diese beiden Werte sind die Skalierungs- und Offsetfaktoren für die Achsenausgabe an die Motorverstärker. Der zweite Wert (Offset) wird von der berechneten Ausgabe (in Volt) subtrahiert und durch den ersten Wert (Skalierungsfaktor) geteilt, bevor er in die D/A-Wandler geschrieben wird. Die Einheiten für den Skalenwert sind in echten Volt pro DAC-Ausgangsspannung. Die Einheiten für den Offset-Wert sind in Volt. Diese können zur Linearisierung eines DAC verwendet werden.

Insbesondere beim Schreiben von Ausgängen, wandelt LinuxCNC zunäschtdie gewünschte Ausgabe von Quasi-SI-Einheiten zu rohen Aktor Werten, z. B. Volt für einen Verstärker DAC. Diese Skalierung sieht wie folgt aus: Der Wert für die Skalierung kann analytisch ermittelt werden, indem man eine Einheitenanalyse durchführt, d.h. die Einheiten sind [Ausgabe-SI-Einheiten]/[Aktoreinheiten]. Bei einer Maschine mit einem Verstärker im Geschwindigkeitsmodus, bei dem 1 Volt zu einer Geschwindigkeit von 250 mm/s führt, ist zu beachten, dass die Einheiten des Offsets in Maschineneinheiten, z. B. mm/s, angegeben sind und von den Sensormesswerten abgezogen werden. Den Wert für diesen Offset erhalten Sie, indem Sie den Wert Ihres Ausgangs ermitteln, der 0,0 für den Aktorausgang ergibt. Wenn der DAC linearisiert ist, so ist dieser Offset normalerweise 0,0.

Skalierung und Offset können auch zur Linearisierung des DAC verwendet werden, so dass sich Werte sich ergeben als Kombination von Verstärkerverstärkung, Nichtlinearität des DAC, DAC-Einheiten usw. . Gehen Sie dazu wie folgt vor:

- Erstellen Sie eine Kalibrierungstabelle für den Ausgang, indem Sie den DAC mit einer gewünschten

Spannung betreiben und das Ergebnis messen:

Table 3. Messungen der Ausgangsspannung

Roh	Gemessen
-10	**-9.93**
-9	**-8.83**
0	**-0.96**
1	**-0.03**
9	**9.87**
10	**10.07**

- Führen Sie eine lineare Anpassung nach dem Prinzip der kleinsten Quadrate durch, um die Koeffizienten a und b so zu ermitteln, dass meas=a*raw+b
- Beachten Sie, dass wir eine Rohausgabe wünschen, bei der das gemessene Ergebnis mit der befohlenen Ausgabe identisch ist. Das bedeutet
  - cmd=a*raw+b
  - raw=(cmd-b)/a
- Folglich können die Koeffizienten a und b aus der linearen Anpassung direkt als Skala und Offset für den Regler verwendet werden.

MAX OUTPUT

Der maximale Wert für den Ausgang der PID-Kompensation, der in den Motorverstärker geschrieben wird, in Volt. Der berechnete Ausgangswert wird auf diesen Grenzwert geklemmt. Der Grenzwert wird vor der Skalierung auf rohe Ausgabeeinheiten angewendet. Der Wert wird symmetrisch sowohl auf die Plus- als auch auf die Minusseite angewandt.

Tuning-Test

Der Abstimmungstest funktioniert leider nur bei schrittmotorbasierten Systemen. Stellen Sie erneut sicher, dass die Richtungen auf der Achse korrekt sind. Dann testen Sie das System, indem Sie die Achse hin und her laufen lassen. Wenn die Beschleunigung oder die maximale Geschwindigkeit zu hoch ist, verlieren Sie Schritte. Beachten Sie beim Hin- und Herlaufen, dass es eine Weile dauern kann, bis eine Achse mit geringer Beschleunigung zum Stillstand kommt. Die Endschalter sind während dieses Tests nicht funktionsfähig. Sie können für jedes Ende der Testbewegung eine Pausenzeit einstellen. Dadurch können Sie eine Messuhr einrichten und ablesen, ob Sie Schritte verlieren.

Schrittmotor-Timing

Das Stepper-Timing muss auf die Anforderungen des Schrittreglers zugeschnitten werden's. PnCconf liefert einige Standard-Timingwerte für den Controller oder erlaubt eigene Timing-Einstellungen. Unter https://wiki.linuxcnc.org/cgi-bin/wiki.pl?Stepper_Drive_Timing finden Sie einige weitere bekannte Timing-Zahlen (Sie können gerne weitere hinzufügen, die Sie selbst

herausgefunden haben). Im Zweifelsfall sollten Sie große Zahlen wie 5000 verwenden, da dies nur die maximale Geschwindigkeit begrenzt.

Bürstenlose Motorsteuerung

Diese Optionen werden verwendet, um eine Low-Level-Steuerung von bürstenlosen Motoren mit spezieller Firmware und Tochterkarten zu ermöglichen. Sie ermöglicht auch die Konvertierung von HALL-Sensoren von einem Hersteller zu einem anderen. Sie wird nur teilweise unterstützt und erfordert, dass man die HAL-Verbindungen fertigstellt. Kontaktieren Sie die Mail-Liste oder das Forum für weitere Hilfe.

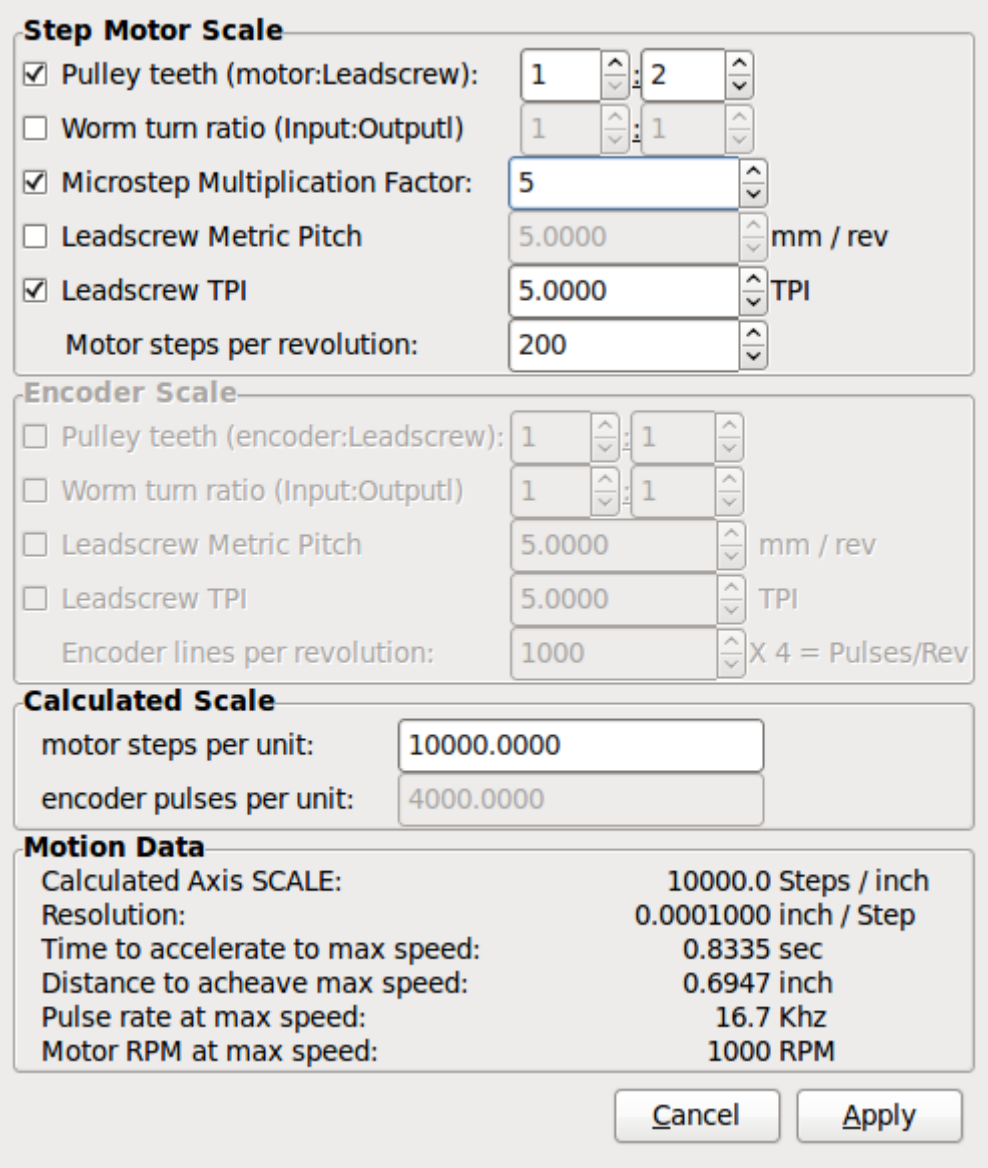


Figure 48. Berechnung der Achsenskala

Die Maßstabseinstellungen können direkt eingegeben werden oder man kann die Schaltfläche *Maßstab berechnen* zur Hilfe nehmen. Verwenden Sie die Kontrollkästchen, um die entsprechenden Berechnungen auszuwählen. Beachten Sie, dass *Riemenscheibenzähne* die Anzahl der Zähne und nicht das Übersetzungsverhältnis erfordert. Das Schneckenradverhältnis ist genau das Gegenteil und erfordert das Zahnradverhältnis. Wenn Sie mit der Skala zufrieden sind, drücken Sie auf Anwenden, andernfalls auf Abbrechen und geben Sie die Skala direkt ein.

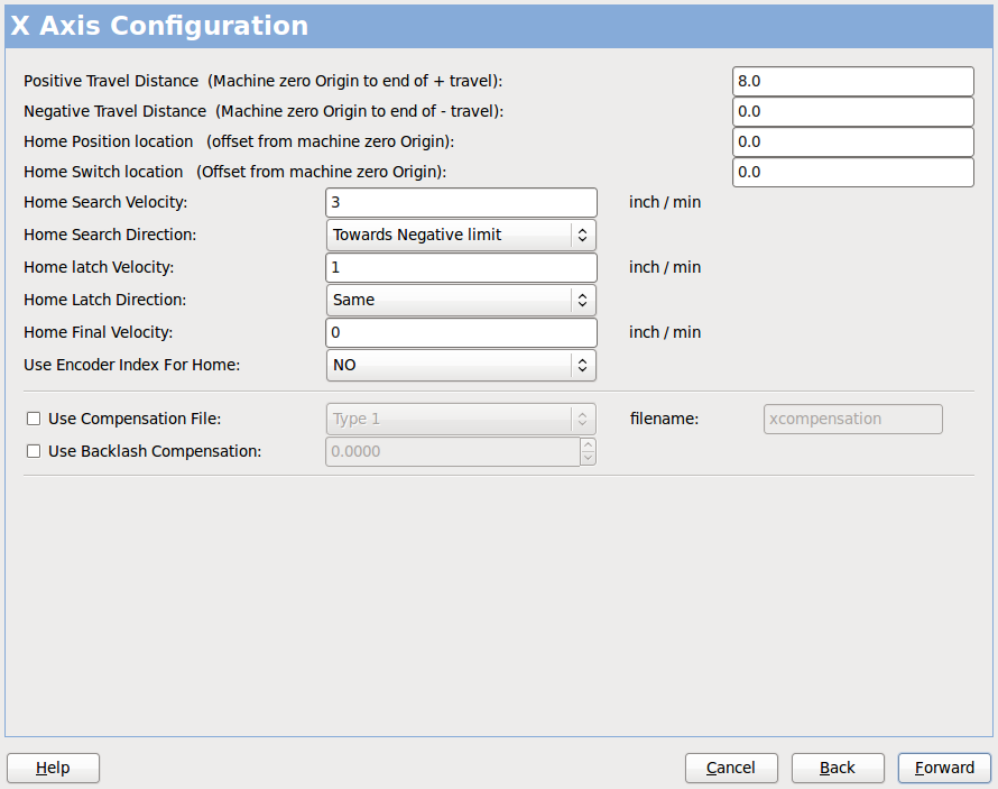


Figure 49. Konfiguration der Achsen

Auf der Registerkarte Diagramm finden Sie außerdem zwei Beispiele für Referenzfahrt- und Endschalter. Dies sind zwei Beispiele für viele verschiedene Möglichkeiten der Einstellung von Referenzfahrt und Grenzwerten.

IMPORTANT Es ist sehr wichtig, dass sich die Achse zu Beginn in die richtige Richtung bewegt, da es sonst sehr schwierig ist, die Referenzfahrt zu richtig durchzuführen!

Denken Sie daran, dass sich positive und negative Richtungen auf das WERKZEUG und nicht auf den Tisch beziehen, wie im Maschinenhandbuch beschrieben.

Bei einer typischen Knie- oder Bettfräse

- Wenn sich die TABLE nach außen bewegt, ist das die positive Y-Richtung
- Wenn sich die TABLE nach links bewegt, ist das die positive X-Richtung
- Wenn sich die TABLE nach unten bewegt, ist das die positive Z-Richtung
- Wenn sich der KOPF nach oben bewegt, ist das die positive Z-Richtung

Bei einer typischen Drehmaschine

- wenn sich das WERKZEUG nach rechts, weg vom Futter, bewegt
- das ist die positive Z-Richtung
- wenn sich das WERKZEUG auf den Bediener zubewegt
- das ist die positive X-Richtung. Einige Drehmaschinen haben eine entgegengesetzte X-Richtung (z.B. Werkzeug auf der Rückseite), das funktioniert gut, aber die grafische Achsen-Anzeige kann nicht so eingestellt werden, dass sie dies widerspiegelt.

Bei der Verwendung von Referenzfahrt- und / oder Endschaltererwartet LinuxCNC die HAL-Signale wahr zu sein, wenn der Schalter gedrückt wird / ausgelöst. Wenn das Signal falsch ist für einen Endschalter dann LinuxCNC wird denken, die Maschine sei die ganze Zeit bereits am Ende der Grenze. Wenn die Referenzfahrt-Schalter Suchlogik falsch ist, wird LinuxCNC den Referenzpunkt scheinbar in der falschen Richtung suchen. Was aber tatsächlich geschieht, ist dass LinuxCNC verusucht, dem mutmaßlich bereits ausgelösten Referenzpunkt-Schalter auszuweichen.

Entscheiden Sie sich für die Position des Endschalters

Endschalter dienen als Backup für Software-Grenzen, falls etwas Elektrisches schief geht, z. B. ein Servo durchdreht. Die Endschalter sollten so platziert werden, dass die Maschine nicht auf das physikalische Ende der Achsenbewegung trifft. Denken Sie daran, dass die Achse bei einer schnellen Bewegung an der Kontaktstelle vorbeilaufen wird. Endschalter sollten *active low* an der Maschine sein. z.B. fließt die ganze Zeit Strom durch die Schalter - ein Stromausfall (offener Schalter) löst aus. Man kann sie zwar auch anders herum verdrahten, aber das ist ausfallsicher. Dies muss möglicherweise invertiert werden, so dass das HAL-Signal in LinuxCNC in *active high* - ein TRUE bedeutet der Schalter ausgelöst wurde. Beim Starten von LinuxCNC, wenn Sie eine On-Limit-Warnung zu bekommen, und die Achse ist NICHT Auslösen des Schalters, Invertieren des Signals ist wahrscheinlich die Lösung. (Verwenden Sie HALMETER, um die entsprechenden HAL-Signal zB joint.0.pos-lim-sw-in X-Achse positive Endschalter zu überprüfen)

Entscheiden Sie sich für den Standort des Referenzpunktschalters

Wenn Sie Endschalter verwenden, können Sie auch einen als Referenzschalter verwenden. Ein separater Referenzpunktschalter ist nützlich, wenn Sie eine lange Achse haben, die in der Regel weit von den Endschaltern entfernt ist, oder wenn das Bewegen der Achse zu den Enden Probleme mit der Beeinträchtigung des Materials mit sich bringt. Hinweis, bei einer langen Welle in einer Drehmaschine ist es schwierig, die Endpunkte anzufahren, ohne dass das Werkzeug die Welle berührt. Wenn Sie einen Drehgeber mit Index haben, dient der Referenzpunktschalter als Referenzpunkt und der Index ist der tatsächliche Referenzpunkt.

Entscheiden Sie sich für die Lage des Maschinen-Ursprungs (engl. MACHINE ORIGIN)

Der MACHINE ORIGIN dient bei LinuxCNC für alle Benutzer-Koordinatensysteme als Referenz. Ich kann mir kaum vorstellen, warum es an einer bestimmten Stelle sein muss. Es gibt nur ein paar G-Codes, um auf die MACHINE COORDINATE System zugreifen können.(G53, G30 und G28) Zusammen mit Werkzeugwechsel-at-G30 Option mit dem Ursprung an der Werkzeugwechselposition kann die praktisch sein. Aus Konvention ist es am einfachsten, den ORIGIN am Referenzpunkt zu haben.

Entscheiden Sie sich für den (endgültigen) Referenzunkt (engl. HOME POSITION)

dies platziert nur den Schlitten an einer konsistenten und bequemen Position nachdem LinuxCNC herausfindet, wo derORIGIN ist.

Messen / Berechnen der positiven / negativen Achsabstände

Fahren Sie die Achse zum Ursprung. Markieren Sie eine Referenz auf dem beweglichen Schlitten und dem unbeweglichen Träger (so dass sie in einer Linie liegen) und fahren Sie die Maschine bis zum Ende der Grenzen. Messen Sie den Abstand zwischen den Markierungen, der einer der Verfahrwege ist. Bewegen Sie den Tisch an das andere Ende des Verfahrwegs. Messen Sie die Markierungen erneut. Das ist der andere Verfahrweg. Wenn sich der URSPRUNG an einer der Begrenzungen befindet, ist dieser Verfahrweg gleich Null.

(Maschinen-)URSPRUNG

Der Ursprung ist der MASCHINENNullpunkt. (nicht der Nullpunkt Sie Ihre Cutter / Material auf). LinuxCNC verwendet diesen Punkt, um alles andere von Referenz. Es sollte innerhalb der Software Grenzen sein. LinuxCNC verwendet die Referenzpunkt (engl. home)-Schalter-Position, um die Ursprungs-Position zu bestimmen (bei Verwendung von Home-Schalter oder muss manuell eingestellt werden, wenn nicht mit Home-Schalter.

Verfahrweg

Dies ist die maximale Entfernung, die eine Achse in jede Richtung fahren kann. Dies kann, muss aber nicht, direkt vom Ursprung bis zum Endschalter gemessen werden. Die positiven und negativen Verfahrwege sollten sich zum Gesamtverfahrweg addieren.

POSITIVER VERFAHRWEG

Dies ist die Entfernung, die auf einer Achse vom Ursprung bis zum positiven Verfahrweg oder dem gesamten Verfahrweg minus dem negativen Verfahrweg zurücklegt wird. Sie würden diesen Wert auf Null setzen, wenn der Ursprung an der positiven Grenze positioniert ist. Der Wert wird immer Null oder eine positive Zahl sein.

NEGATIVER VERFAHRWEG (engl. travel distance)

Dies ist die Entfernung, die auf einer Achse vom Ursprung bis zum negativen Verfahrweg zurücklegt werden kann oder der gesamte Verfahrweg minus dem positiven Verfahrweg. Sie würden diesen Wert auf Null setzen, wenn der Ursprung an der negativen Grenze positioniert ist. Dieser Wert ist immer Null oder eine negative Zahl. Wenn Sie vergessen, diesen Wert negativ zu setzen, wird dies von PnCconf intern erledigt.

(Letzlicher) REFERENZPUNKT (engl. home position)

Dies ist die Position, an der die Startsequenz enden wird. Sie bezieht sich auf den Ursprung, kann also negativ oder positiv sein, je nachdem, auf welcher Seite des Ursprungs sie sich befindet. Wenn Sie sich an der (endgültigen) Ausgangsposition befinden und sich in positiver Richtung bewegen müssen, um zum Ursprung zu gelangen, wird die Zahl negativ sein.

Referenzpunkt-Schalter Position

Dies ist der Abstand zwischen dem Home-Schalter und dem Ursprung (engl. origin). Sie kann negativ oder positiv sein, je nachdem, auf welcher Seite des Ursprungs sie sich befindet. Wenn Sie sich an der Position des Home-Schalters in positiver Richtung bewegen müssen, um zum Ursprung zu gelangen, ist die Zahl negativ. Wenn Sie diesen Wert auf Null setzen, befindet sich der Ursprung an der Position des Endschalters (plus Entfernung zum Index, falls verwendet).

Referenzpunkt Suchgeschwindigket (engl. home search velocity)

Geschwindigkeit bei der Suche nach dem Kursziel in Einheiten pro Minute.

Referenzpunkt-Suchrichtung (engl. home search direction)

Legt die Suchrichtung des Referenzschalters entweder negativ (d. h. in Richtung des negativen Endschalters) oder positiv (d. h. in Richtung des positiven Endschalters) fest.

Referenzpunkt Latch Geschwindigkeit

Feinfühlige Home-Suchgeschwindigkeit in Einheiten pro Minute.

Referenzpunktsuche minimale Geschwindigkeit (engl. Home Final Velocity)

Geschwindigkeit von der latch-Position zur (endgültigen) Ausgangsposition in Einheiten pro Minute. Für maximale Eilgeschwindigkeit auf 0 setzen.

Referenzpunkt der Verriegelungsrichtung

Ermöglicht die Einstellung der Verriegelungsrichtung auf die gleiche oder entgegengesetzte Richtung wie die Suchrichtung.

Encoder-Index für Referenzpunkt verwenden

LinuxCNC sucht während der Latch-Phase der Referenzfahrt nach einem Encoder-Indeximpuls.

Kompensationsdatei verwenden

Ermöglicht die Angabe eines Komp-Dateinamens und -typs. Ermöglicht eine anspruchsvolle Kompensation. Siehe den Achsen-Abschnitt des INI Kapitels.

Verwenden des Umkehrspiel-Ausgleichs

Ermöglicht die Einstellung einer einfachen Kompensation des Umkehrspiels. Kann nicht mit Kompensationsdatei verwendet werden. Siehe den Achsen-Abschnitt des INI Kapitels.

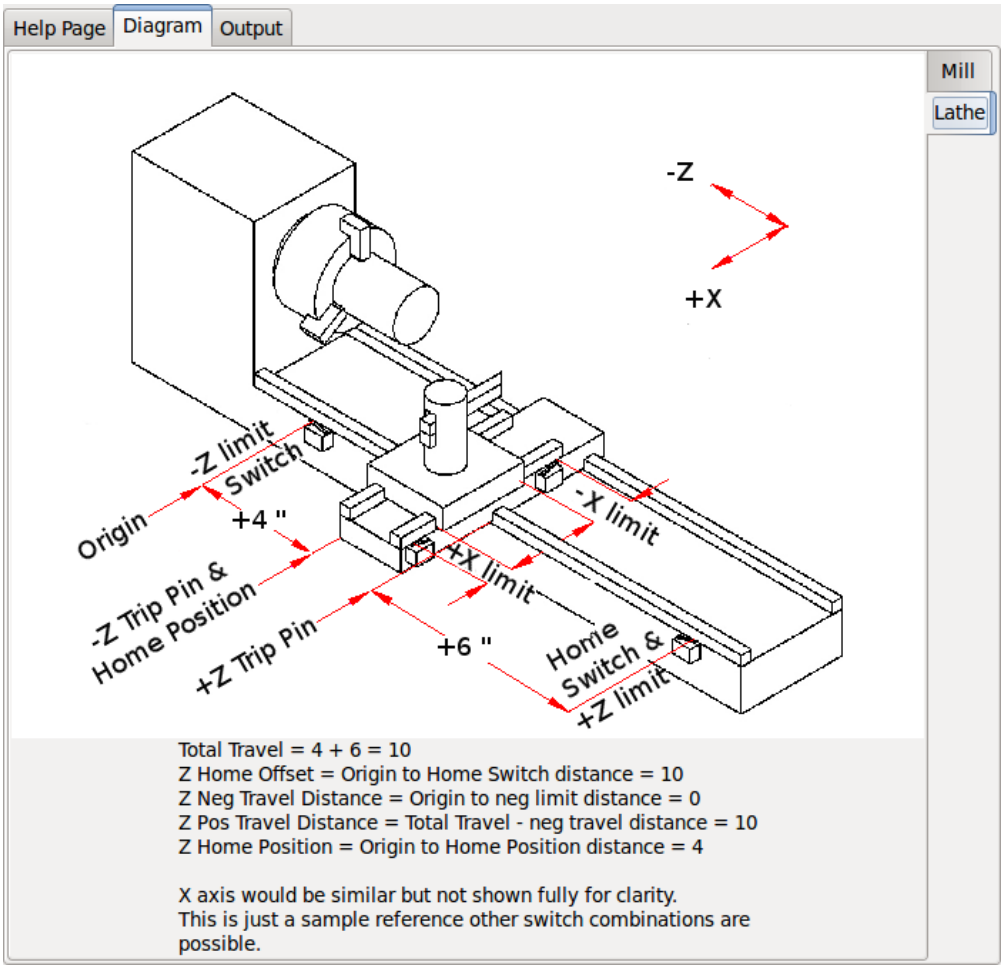


Figure 50. AXIS-Hilfsdiagramm

Das Diagramm soll helfen, ein Beispiel für Endschalter und Standard-Achsbewegungsrichtungen zu demonstrieren. In diesem Beispiel wurde die Z-Achse mit zwei Endschaltern versehen, wobei der positive Schalter als Home-Schalter verwendet wird. Der Maschinen-Ursprung (Nullpunkt, engl.

machine origin) befindet sich am negativen Endschalter. Die linke Kante des Schlittens ist der negative Grenzwert und die rechte der positive Grenzwert. Die ENDGÜLTIGE HOME-POSITION soll 4 Zoll vom ORIGIN auf der positiven Seite entfernt sein. Wenn der Schlitten an die positive Grenze bewegt würde, würden wir 10 Zoll zwischen der negativen Grenze und dem negativen Auslösestift messen.

3.2.10. Spindel-Konfiguration

Wenn Sie Spindelsignale auswählen, ist diese Seite zur Konfiguration der Spindelsteuerung verfügbar.

TIP Viele der Optionen auf dieser Seite werden nur angezeigt, wenn auf den vorherigen Seiten die richtige Option ausgewählt wurde!

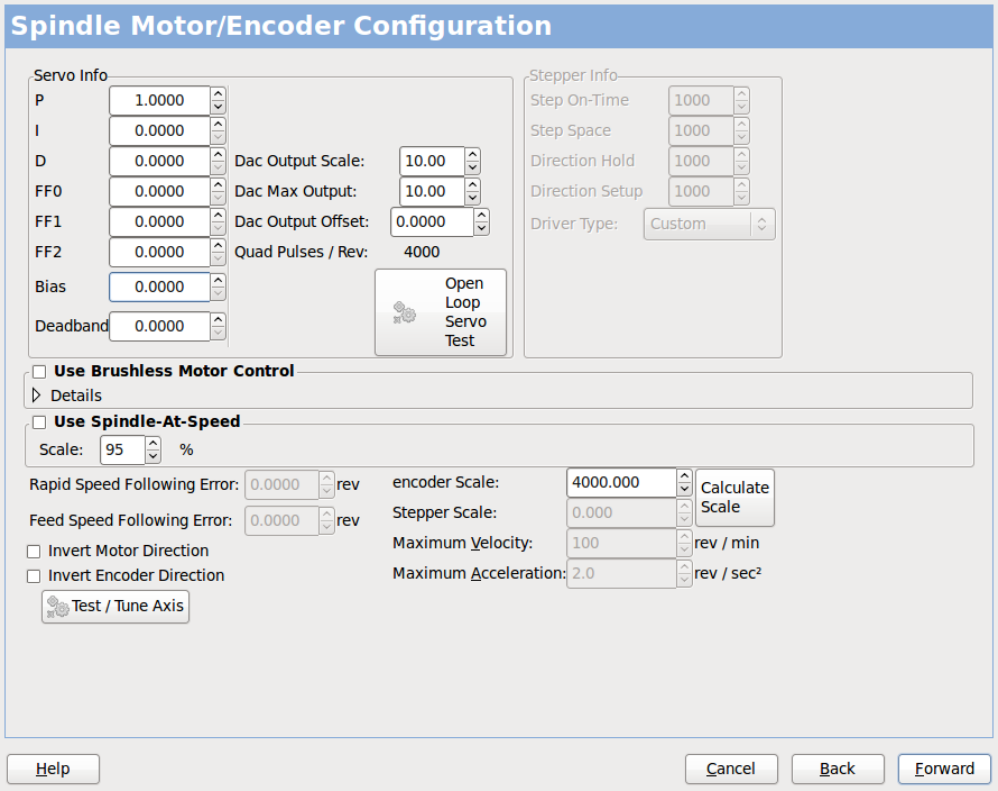


Figure 51. Spindelmotor/Encoder-Konfiguration

Diese Seite ähnelt der Seite zur Konfiguration der Achsenmotoren.

Es gibt einige Unterschiede:

- Sofern man sich nicht für eine schrittgetriebene Spindel entschieden hat, gibt es keine Beschleunigungs- oder Geschwindigkeitsbegrenzung.
- Es gibt keine Unterstützung für Gangschaltungen oder Bereiche.
- Wenn Sie eine VCP-Spindelanzeigeoption gewählt haben, können die Skala für die Spindeldrehzahl und die Filtereinstellungen angezeigt werden.
- Spindle-at-Speed ermöglicht LinuxCNC zu warten, bis die Spindel auf die gewünschte Geschwindigkeit vor dem Bewegen der Achse ist. Dies ist besonders praktisch auf Drehmaschinen mit konstantem Oberflächenvorschub und große Geschwindigkeit Durchmesseränderungen. Es erfordert entweder Encoder-Feedback oder eine digitale Spindel-at-Speed-Signal in der Regel zu

einem VFD-Antrieb verbunden.

- Wenn Sie eine Encoder-Rückmeldung verwenden, können Sie eine Skaleneinstellung für die Spindeldrehzahl wählen, die angibt, wie nahe die tatsächliche Drehzahl an der geforderten Drehzahl liegen muss, damit sie als gleichbleibende Drehzahl gilt.
- Bei Verwendung von Encoder-Feedback kann die VCP-Drehzahlanzeige unregelmäßig sein - die Filtereinstellung kann zur Glättung der Anzeige verwendet werden. Die Geberskala muss für die verwendete Geberzahl/Getriebe eingestellt werden.
- Wenn Sie einen einzelnen Eingang für einen Spindel-Drehgeber verwenden, müssen Sie die Zeile: setp hm2_7i43.0.encoder.00.counter-mode 1 (wobei Sie den Namen der Karte und die Nummer des Drehgebers entsprechend Ihren Anforderungen ändern) in eine benutzerdefinierte HAL-Datei einfügen. Weitere Informationen zum Zählermodus finden Sie im Abschnitt Encoder in Hostmot2.

3.2.11. Weitere Optionen für Fortgeschrittene

Dies ermöglicht die Einstellung von HALUI-Befehlen und das Laden von ClassicLadder- und Beispiel-SPS-Programme. Wenn Sie GladeVCP-Optionen ausgewählt haben, z. B. zum Nullstellen der Achse, werden Befehle angezeigt. Im Kapitel HALUI finden Sie weitere Informationen zur Verwendung benutzerdefinierter halcmds. Es gibt mehrere Optionen für Kontaktplanprogramme. Das Notaus (engl. E-stop)-Programm ermöglicht es einem externen Notaus-Schalter oder dem GUI-Frontend, ein Notaus auszulösen. Es verfügt auch über ein zeitgesteuertes Schmiermittelpumpensignal. Das Z-Auto-Touch-Off-Programm verfügt über eine Touch-Off-Platte, die GladeVCP-Touch-Off-Taste und spezielle HALUI-Befehle, um den aktuellen Benutzerursprung auf Null zu setzen und schnell zu löschen. Das serielle Modbus-Programm ist im Grunde eine leere Programmvorlage, die ClassicLadder für seriellen Modbus einrichtet. Siehe das Kapitel ClassicLadder im Handbuch.

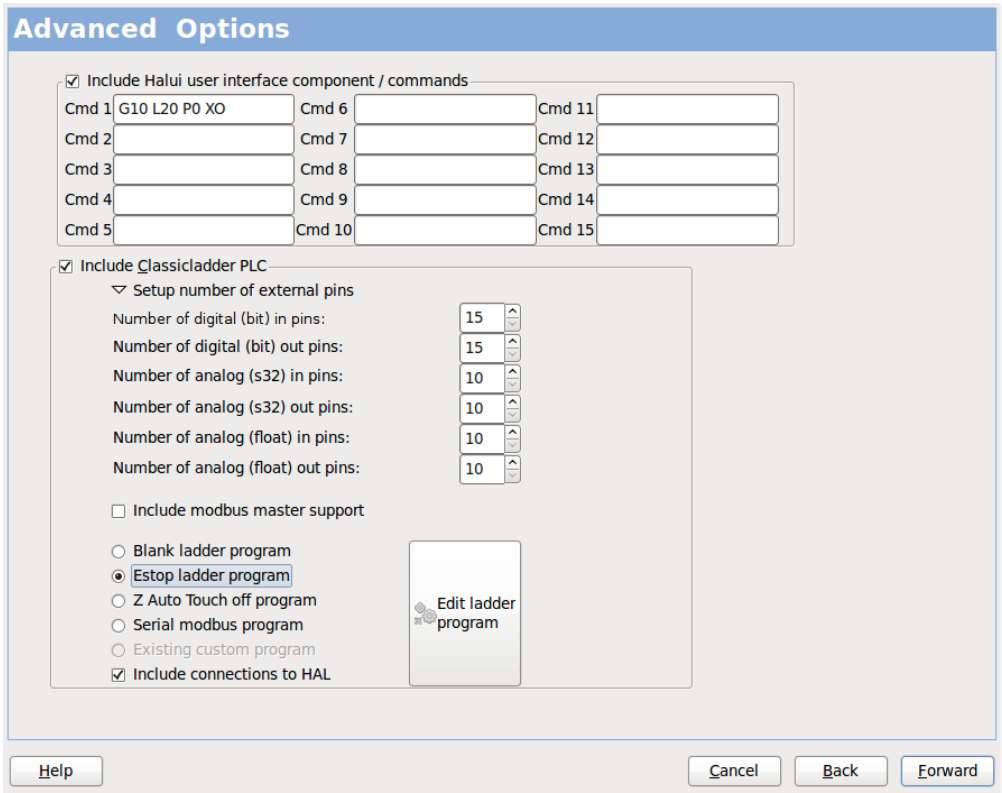


Figure 52. PnCconf, erweiterte Optionen

3.2.12. HAL-Komponenten

Auf dieser Seite können Sie zusätzliche HAL-Komponenten hinzufügen, die Sie für benutzerdefinierte HAL-Dateien benötigen. Auf diese Weise sollte man die Haupt-HAL-Datei nicht von Hand bearbeiten müssen, aber dennoch die vom Benutzer benötigten Komponenten bei der Konfiguration berücksichtigen.

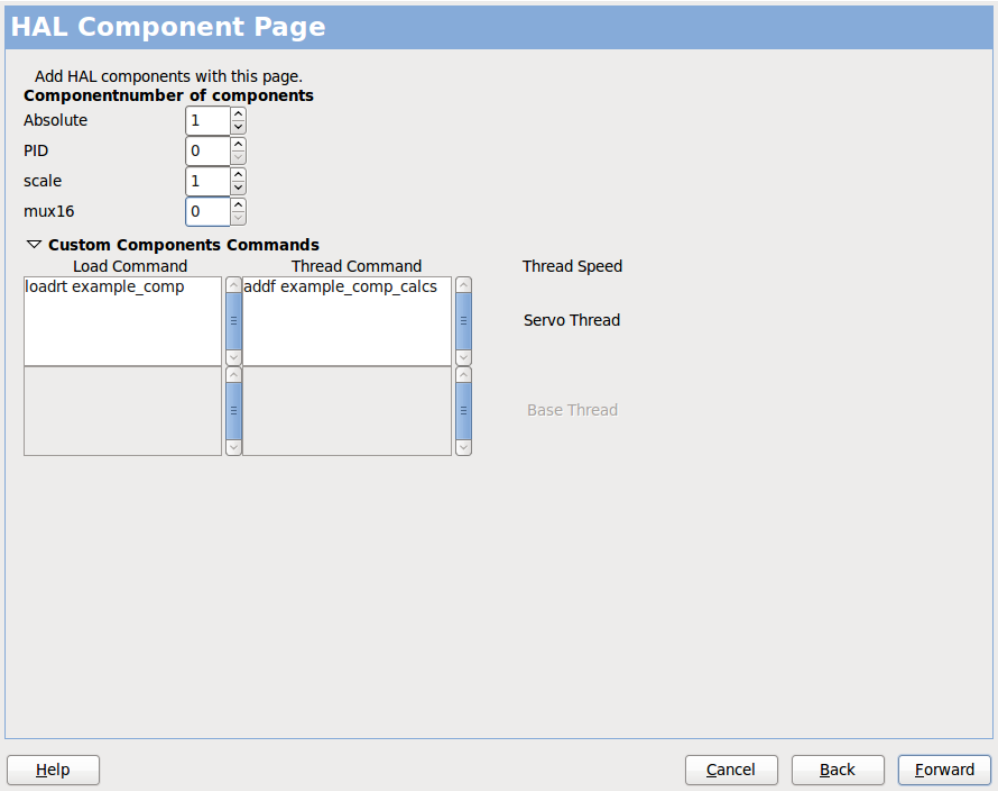


Figure 53. HAL-Komponenten

Die erste Auswahl sind Komponenten, die PnCconf intern verwendet. Sie können pncconf so konfigurieren, dass zusätzliche Instanzen der Komponenten für Ihre eigene HAL-Datei geladen werden.

Wählen Sie die Anzahl der Instanzen, die Ihre benutzerdefinierte Datei benötigt, PnCconf fügt die benötigten Instanzen danach hinzu.

Das heißt, wenn Sie 2 benötigen und PnCconf 1 benötigt, wird PnCconf 3 Instanzen laden und die letzte verwenden.

Benutzerdefinierte Komponenten-Befehle

Mit dieser Auswahl können Sie HAL-Komponenten laden, die PnCconf nicht verwendet. Fügen Sie den Befehl loadrt oder loadusr unter der Überschrift *loading command* hinzu. Fügen Sie den Befehl addf unter der Überschrift *Thread-Befehl* hinzu. Die Komponenten werden dem Thread zwischen dem Lesen von Eingaben und dem Schreiben von Ausgaben in der Reihenfolge hinzugefügt, in der Sie sie im Befehl "thread" schreiben.

3.2.13. PnCconf für Fortgeschrittene

PnCconf ist bestrebt, flexible Anpassungen durch den Benutzer zu ermöglichen. PnCconf unterstützt benutzerdefinierte Signalnamen, benutzerdefiniertes Laden von Komponenten, benutzerdefinierte

HAL-Dateien und benutzerdefinierte Firmware.

Es gibt auch Signalnamen, die PnCconf immer bereitstellt, unabhängig von den gewählten Optionen für benutzerdefinierte HAL-Dateien. Mit etwas Überlegung sollten die meisten Anpassungen funktionieren, auch wenn Sie später andere Optionen in PNCconf wählen.

Wenn die Anpassungen den Rahmen von PnCconf's Rahmenwerk sprengen, können Sie PnCconf verwenden, um eine Basiskonfiguration zu erstellen, oder Sie verwenden eine der LinuxCNC's Beispielkonfigurationen und editieren sie von Hand zu dem, was Sie wollen.

Benutzerdefinierte Signalnamen

Wenn Sie eine Komponente mit etwas in einer benutzerdefinierten HAL-Datei verbinden möchten, geben Sie einen eindeutigen Signalnamen in das Kombinationsfeld ein. Bestimmte Komponenten fügen Endungen an Ihren benutzerdefinierten Signalnamen an:

Kodierer fügen hinzu < customname > +:

- Position
- count (engl für Zähler)
- Geschwindigkeit
- Index-Aktivierung
- reset

Schrittmotoren fügen hinzu:

- aktivieren
- Zähler
- Positionsbefehl
- position-fb
- velocity-fb

Pulsweitenmodulationen (PWM, für Servos) fügen hinzu:

- aktivieren
- Wert

GPIO-Pins werden einfach mit dem eingegebenen Signalnamen verbunden

Auf diese Weise kann man sich mit diesen Signalen in den benutzerdefinierten HAL-Dateien verbinden und hat trotzdem die Möglichkeit, sie später zu verschieben.

Benutzerdefinierte Signalnamen

Die Seite mit HAL Komponenten kann verwendet werden, um Komponenten zu laden, die ein Benutzer für die Anpassung benötigt.

Laden der benutzerdefinierten Firmware

PnCconf sucht auf dem System nach Firmware und sucht dann nach der XML-Datei, die es in das konvertieren kann, was es versteht. Diese XML-Dateien werden nur für offiziell freigegebene Firmware vom LinuxCNC-Team bereitgestellt. Um benutzerdefinierte Firmware zu verwenden, muss man sie in ein Array konvertieren, das PnCconf versteht, und den Dateipfad zu PnCconf's Einstellungsdatei hinzufügen. Standardmäßig sucht dieser Pfad auf dem Desktop nach einem Ordner namens custom_firmware und einer Datei namens firmware.py.

Die versteckte Einstellungsdatei befindet sich in der Home-Datei des Benutzers, heißt .pncconf-preferences und erfordert, dass Sie in Ihrem Dateimanager die Option "Versteckte Dateien anzeigen" wählen, um sie zu sehen und zu bearbeiten, oder Sie verwenden auf der Kommandozeile "ls" mit der Option "-a". Der Inhalt dieser Datei kann eingesehen werden, wenn Sie PnCconf zum ersten Mal laden - drücken Sie die Hilfetaste und sehen Sie sich die Ausgabeseite an.

Fragen Sie in der LinuxCNC Mailing-Liste oder im Forum nach Informationen über die Konvertierung von kundenspezifischer Firmware. Nicht jede Firmware kann mit PnCconf verwendet werden.

Benutzerdefinierte HAL-Dateien

Es gibt vier benutzerdefinierte Dateien, die Sie verwenden können, um HAL-Befehle hinzuzufügen:

- custom.hal ist für HAL-Befehle, die nicht nach dem Laden des GUI-Frontends ausgeführt werden müssen. Es wird diese erst nach der HAL-Datei mit dem Konfigurationsnamen ausgeführt.
- custom_postgui.hal ist für Befehle gedacht, die ausgeführt werden müssen, nachdem AXIS geladen wurde oder eine eigenständige PyVCP-Anzeige geladen wurde.
- custom_gvcp.hal ist für Befehle, die ausgeführt werden müssen, nachdem GladeVCP geladen wurde.
- shutdown.hal ist für Befehle, die ausgeführt werden, wenn LinuxCNC kontrolliert herunterfährt.

Chapter 4. Konfiguration

4.1. Integrator-Konzepte

4.1.1. Dateispeicherorte

LinuxCNC sucht nach den Konfigurations- und G-Code-Dateien an einem bestimmten Ort. Der Ort hängt davon ab, wie Sie LinuxCNC ausführen.

Installiert

Wenn Ihr LinuxCNC von der Live-CD oder Sie über eine .deb installiert haben, und verwenden Sie die Konfiguration Picker *LinuxCNC* aus dem Menü LinuxCNC, so schaut LinuxCNC in die folgenden Verzeichnisse:

- Das LinuxCNC-Verzeichnis befindet sich unter ‚/home/benutzername/linuxcnc‘.
- Die Konfigurationsverzeichnisse befinden sich unter */home/benutzername/linuxcnc/configs*.
  - Die Konfigurationsdateien befinden sich unter */home/benutzername/linuxcnc/configs/name-of-config*.
- Die G-Code-Dateien befinden sich unter /home/benutzername/linuxcnc/nc_files'.

Bei einer Konfiguration mit dem Namen Mill und dem Benutzernamen Fred würde die Verzeichnis- und Dateistruktur zum Beispiel wie folgt aussehen.

- */home/fred/linuxcnc*
- */home/fred/linuxcnc/nc_files*
- */home/fred/linuxcnc/configs/mill*
  - */home/fred/linuxcnc/configs/mill/mill.ini*
  - */home/fred/linuxcnc/configs/mill/mill.hal*
  - */home/fred/linuxcnc/configs/mill/mill.var*
  - */home/fred/linuxcnc/configs/mill/tool.tbl*

Befehlszeile

Wenn Sie LinuxCNC von der Kommandozeile aus und geben Sie den Namen und den Speicherort der INI-Datei können die Dateispeicherorte in einem anderen Ort sein. Um die Optionen für die Ausführung von LinuxCNC von der Kommandozeile laufen *linuxcnc -h*.

NOTE	Optionale Speicherorte für einige Dateien können in der INI-Datei konfiguriert werden. Siehe den Abschnitt `<<sub:ini:sec:display,[DISPLAY]>>` und den Abschnitt `<<sub:ini:sec:rs274ngc,[RS274NGC]>>`.

4.1.2. Dateien

Jedes Konfigurationsverzeichnis benötigt mindestens die folgenden Dateien:

- Eine INI-Datei .ini
- Eine HAL-Datei .hal oder HALTCL-Datei .tcl, die im Abschnitt HAL der INI-Datei angegeben ist.

NOTE Für einige GUIs können andere Dateien erforderlich sein.

Optional können Sie auch haben:

- Eine Variablendatei .var
  - Wenn Sie eine .var-Datei in einem Verzeichnis weglassen, aber `<<sub:ini:sec:rs274ngc,[RS274NGC]>>` PARAMETER_FILE=somefilename.var, wird die Datei für Sie erstellt werden, wenn LinuxCNC startet.
  - Wenn Sie eine .var-Datei weglassen und den Punkt [RS274NGC] PARAMETER_FILE weglassen, wird eine var-Datei mit dem Namen rs274ngc.var erstellt, wenn LinuxCNC startet. Es kann einige verwirrende Meldungen geben, wenn [RS274NGC]PARAMETER_FILE weggelassen wird.
- Eine Werkzeugtabellendatei .tbl, wenn `<<sub:ini:sec:emcmot,[EMCMOT]>>` TOOL_TABLE in der INI-Datei angegeben wurde. Einige Konfigurationen benötigen keine Werkzeugtabelle.

4.1.3. Schrittmotor-Systeme (engl. stepper systems)

Basiszeitraum (engl. base period)

BASE_PERIOD ist der *Herzschlag* von Ihrem LinuxCNC Computer.[1]. In jeder Periode entscheidet der Software-Schrittgenerator, ob es Zeit für einen weiteren Schrittimpuls ist. Eine kürzere Periode ermöglicht es Ihnen, mehr Impulse pro Sekunde zu erzeugen, innerhalb von Grenzen. Wenn Sie jedoch eine zu kurze Periode wählen, verbringt Ihr Computer so viel Zeit mit der Erzeugung von Schrittimpulsen, dass alles andere langsamer wird oder vielleicht sogar zum Stillstand kommt. Die Latenzzeit und die Anforderungen an die Schrittmotorsteuerung beeinflussen die kürzeste Zeitspanne, die Sie verwenden können.

Im schlimmsten Fall treten Latenzzeiten nur ein paar Mal pro Minute auf und die Wahrscheinlichkeit, dass eine schlechte Latenz genau dann auftritt, wenn der Motor die Richtung ändert, ist gering. Es kann also zu sehr seltenen Fehlern kommen, die hin und wieder ein Teil ruinieren und bei denen eine Fehlerbehebung unmöglich ist.

Am einfachsten lässt sich dieses Problem vermeiden, indem Sie eine BASE_PERIOD wählen, die der Summe aus der längsten Zeitanforderung Ihres Laufwerks und der schlimmsten Latenz Ihres Computers entspricht. Dies ist nicht immer die beste Wahl. Wenn Sie z. B. ein Laufwerk mit einer Haltezeit von 20 µs für das Richtungssignal betreiben und Ihr Latenztest eine maximale Latenz von 11 µs angibt, erhalten Sie, wenn Sie die BASE_PERIOD auf 20+11 = 31 µs einstellen, in einem Modus 32.258 Schritte pro Sekunde und in einem anderen Modus 16.129 Schritte pro Sekunde, was nicht gerade angenehm ist.

Das Problem liegt in der erforderlichen Haltezeit von 20 µs. Das plus die 11 µs Latenz ist, was uns zwingt,

eine langsame 31 µs Zeitraum zu verwenden. Aber die LinuxCNC Software Schritt-Generator hat einige Parameter, mit denen Sie die verschiedenen Zeiten von einer Periode auf mehrere zu erhöhen. Zum Beispiel, wenn *steplen* Fußnote:[steplen bezieht sich auf einen Parameter, der die Leistung von LinuxCNC eingebauten Schritt-Generator, *stepgen*, die eine HAL-Komponente ist einstellt. Dieser Parameter passt die Länge des Schrittimpulses selbst. Lesen Sie weiter, alles wird schließlich erklärt werden.] von 1 auf 2 geändert wird, dann wird es zwei Perioden zwischen dem Beginn und dem Ende des Schrittimpulses sein. Wird *dirhold* [2] von 1 auf 3 geändert, dann liegen mindestens drei Perioden zwischen dem Schrittimpuls und einem Wechsel des Richtungspins.

Wenn wir "dirhold" verwenden können, um die 20-µs-Haltezeitanforderung zu erfüllen, dann ist die nächstlängere Zeit die 4,5-µs-High-Time. Addiert man die Latenzzeit von 11 µs zu der hohen Zeit von 4,5 µs, so erhält man eine Mindestzeit von 15,5 µs. Wenn Sie 15,5 µs ausprobieren, stellen Sie fest, dass der Computer zu träge ist, also entscheiden Sie sich für 16 µs. Wenn wir "dirhold" auf 1 lassen (Standardeinstellung), dann ist die Mindestzeit zwischen Schritt und Richtung die 16 µs Periode minus die 11 µs Latenz = 5 µs , was nicht genug ist. Wir brauchen weitere 15 µs . Da die Periode 16 µs beträgt, brauchen wir eine weitere Periode. Also ändern wir *dirhold* von 1 auf 2. Jetzt beträgt die Mindestzeit zwischen dem Ende des Schrittimpulses und dem Richtungswechsel 5+16=21 µs, und wir müssen uns keine Sorgen mehr machen, dass der Antrieb aufgrund der Latenz die falsche Richtung einschlägt.

Weitere Informationen zu stepgen finden Sie im Abschnitt stepgen.

Schritt-Timing

Schrit-Timing und Schrittweite sind bei einigen Antrieben unterschiedlich. In diesem Fall wird der Schrittpunkt wichtig. Wenn der Antrieb bei der fallenden Flanke schaltet, sollte der Ausgangspin invertiert werden.

4.1.4. Servosysteme

Grundbetrieb

Servosysteme sind in der Lage, eine höhere Geschwindigkeit und Genauigkeit zu erreichen als entsprechende Schrittmachersysteme, sind aber teurer und komplexer. Im Gegensatz zu Schrittmotorensystemen benötigen Servosysteme eine Art von Positionsrückmeldung und müssen eingestellt oder getunt werden, da sie nicht wie Schrittmotorensysteme direkt nach dem Auspacken funktionieren. Diese Unterschiede bestehen, weil Servos ein *geschlossener Regelkreis* sind, im Gegensatz zu Schrittmotoren, die im Allgemeinen *offener Regelkreis* betrieben werden. Was bedeutet *geschlossener Regelkreis*? Schauen wir uns ein vereinfachtes Diagramm an, wie ein Servomotorensystem angeschlossen ist.

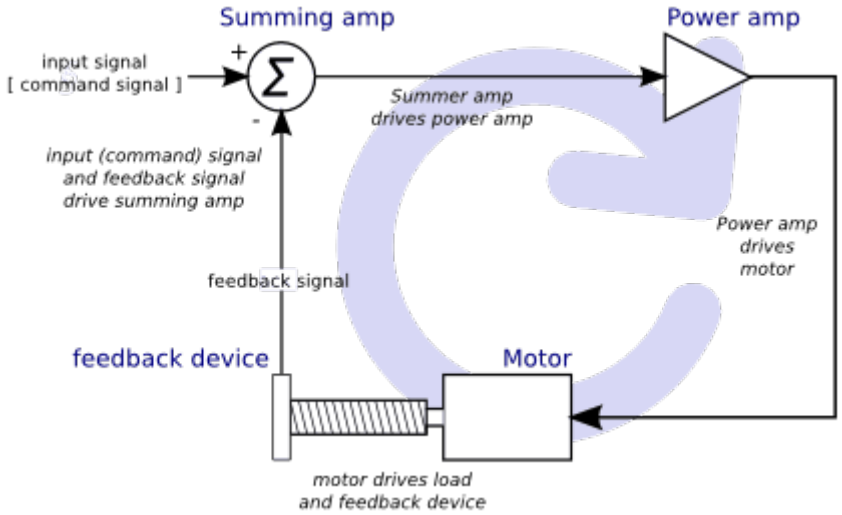


Figure 54. Servo Loop

Dieses Diagramm zeigt, dass das Eingangssignal (und das Rückkopplungssignal) den Summierverstärker antreibt, der Summierverstärker den Leistungsverstärker antreibt, der Leistungsverstärker den Motor antreibt, der Motor die Last (und das Rückkopplungsgerät) antreibt und das Rückkopplungsgerät (und das Eingangssignal) den Motor antreibt. Dies sieht aus wie ein Kreis (eine geschlossene Schleife), in dem A B, B C, C D und D A steuert.

Wenn Sie bisher noch nicht mit Servosystemen gearbeitet haben, wird Ihnen das zweifellos zunächst sehr seltsam vorkommen, vor allem im Vergleich zu normalen elektronischen Schaltungen, bei denen die Eingänge nahtlos zu den Ausgängen führen und nicht zurück.Fußnote:[Falls es hilft, das nächstliegende Äquivalent in der digitalen Welt sind *Zustandsmaschinen*, *sequentielle Maschinen* und so weiter, wo das, was die Ausgänge *jetzt* tun, davon abhängt, was die Eingänge (und die Ausgänge) *vorher* getan haben. Wenn das nicht hilft, dann ist es eben so]. Wenn *alles* steuert *alles andere*, wie kann das jemals funktionieren, wer ist verantwortlich? Die Antwort ist, dass LinuxCNC 'kann dieses System steuern, aber es muss es durch die Wahl einer von mehreren Kontrollmethoden zu tun. Die Steuerungsmethode, die LinuxCNC verwendet, eine der einfachsten und besten, wird PID genannt.

PID steht für Proportional, Integral und Derivativ. Der Proportionalwert bestimmt die Reaktion auf den aktuellen Fehler, der Integralwert bestimmt die Reaktion auf der Grundlage der Summe der letzten Fehler und der Derivativwert bestimmt die Reaktion auf der Grundlage der Rate, mit der sich der Fehler geändert hat. Sie sind drei gemeinsame mathematische Techniken, die auf die Aufgabe, einen Arbeitsprozess, um einen Sollwert zu folgen angewendet werden. Im Fall von LinuxCNC ist der Prozess, den wir steuern wollen, die tatsächliche Achsenposition und der Sollwert ist die befohlene Achsenposition.

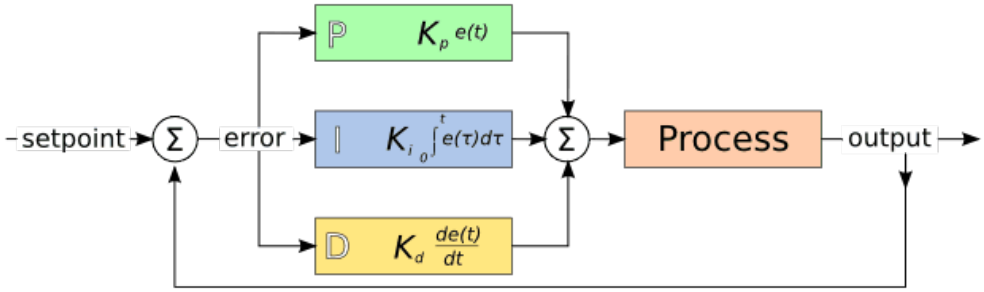


Figure 55. PID-Schleife

Durch *Abstimmung* der drei Konstanten im PID-Regler-Algorithmus kann der Regler eine auf die spezifischen Prozessanforderungen abgestimmte Regelwirkung erzielen. Die Reaktion des Reglers lässt sich beschreiben anhand des Ansprechens des Reglers auf eine Regelabweichung, des Ausmaßes, in dem der Regler über den Sollwert hinausschießt, und des Grades der Systemschwingung.

Proportionaler Ausdruck

Der proportionale Ausdruck(manchmal als Verstärkung bezeichnet) nimmt eine Änderung am Ausgang vor, die proportional zum aktuellen Fehlerwert ist. Eine hohe proportionale Verstärkung führt zu einer großen Änderung des Ausgangs bei einer gegebenen Änderung des Fehlers. Wenn die Proportionalverstärkung zu hoch ist, kann das System instabil werden. Im Gegensatz dazu führt eine kleine Verstärkung zu einer kleinen Ausgangsantwort auf einen großen Eingangsfehler. Wenn die Proportionalverstärkung zu niedrig ist, kann der Regeleingriff bei der Reaktion auf Netzstörungen zu gering sein.

Bei Abwesenheit von Störungen pendelt sich eine reine Proportionalregelung nicht auf ihren Zielwert ein, sondern behält einen stationären Fehler bei, der eine Funktion der Proportionalverstärkung und der Prozessverstärkung ist. Trotz des stationären Offsets zeigen sowohl die Abstimmungstheorie als auch die industrielle Praxis, dass der Proportionalanteil den größten Teil der Ausgangsänderung ausmachen sollte.

Integraler Begriff

Der Beitrag des Integral-Anteils (manchmal im Englischen auch Reset genannt, oder kurz I-Anteil) ist proportional zur Größe des Fehlers und zur Dauer des Fehlers. Die Summierung des momentanen Fehlers über die Zeit (Integration des Fehlers) ergibt die akkumulierte Abweichung, die zuvor hätte korrigiert werden müssen. Der kumulierte Fehler wird dann mit der Integralverstärkung multipliziert und zum Reglerausgang addiert.

Der Integral-Anteil (wenn er zum Proportional-Anteil (kurz P-Anteil) hinzugefügt wird) beschleunigt die Bewegung des Prozesses in Richtung Sollwert und beseitigt den verbleibenden stationären Fehler, der bei einem reinen Proportionalregler auftritt. Da der Integral-Anteil jedoch auf akkumulierte Fehler aus der Vergangenheit reagiert, kann er dazu führen, dass der aktuelle Wert über den Sollwert hinausschießt (den Sollwert überschreitet und dann eine Abweichung in die andere Richtung erzeugt).

Differenzierender-Anteil (D-Anteil)

Die Änderungsrate des Prozessfehlers wird berechnet, indem die Steigung des Fehlers nach der Zeit (d. h. seine erste Ableitung nach der Zeit) bestimmt und diese Änderungsrate mit der Ableitungsverstärkung multipliziert wird.

Der Derivationsanteil verlangsamt die Änderungsrate des Reglerausgangs, und dieser Effekt ist in der Nähe des Reglersollwerts am deutlichsten. Daher wird die Ableitungsregelung eingesetzt, um das Ausmaß des durch den Integralanteil verursachten Überschwingens zu verringern und die kombinierte Regler-Prozess-Stabilität zu verbessern.

Schleifenabstimmung

Wenn die Parameter des PID-Reglers (die Verstärkungen des Proportional-, Integral- und Differentialanteils) falsch gewählt werden, kann der geregelte Prozesseingang instabil sein, d. h. sein Ausgang divergiert, mit oder ohne Schwingung, und wird nur durch Sättigung oder mechanischen Bruch begrenzt. Die Abstimmung eines Regelkreises ist die Anpassung seiner Regelparameter (Verstärkung/Proportionalbereich, Integralverstärkung/Rückstellung, Ableitungsverstärkung/Rate) an die optimalen Werte für das gewünschte Regelverhalten.

Manuelle Abstimmung

Eine einfache Abstimmungsmethode besteht darin, zunächst die Werte I und D auf Null zu setzen. Erhöhen Sie den P-Wert, bis das Ausgangssignal der Schleife oszilliert, dann sollte der P-Wert auf etwa die Hälfte dieses Wertes eingestellt werden, um eine Reaktion vom Typ *Viertelamplitudenabfall* zu erzielen. Erhöhen Sie dann I, bis der Offset in ausreichender Zeit für den Prozess korrigiert ist. Ein zu großer I-Wert führt jedoch zu Instabilität. Erhöhen Sie schließlich D, falls erforderlich, bis die Schleife nach einer Laststörung akzeptabel schnell ihren Sollwert erreicht. Ein zu großes D führt jedoch zu übermäßigem Ansprechen und Überschwingen. Eine schnelle PID-Schleifenabstimmung schwingt in der Regel leicht über, um den Sollwert schneller zu erreichen; einige Systeme können jedoch kein Überschwingen akzeptieren, in diesem Fall ist ein *überdämpftes* Regelsystem erforderlich, das eine P-Einstellung erfordert, die deutlich unter der Hälfte der P-Einstellung liegt, die eine Schwingung verursacht.

4.1.5. S-Kurven (engl. S-curve)Trajektorienplanung

S-curve trajectory planning limits jerk (the rate of change of acceleration) to provide smoother motion. This can reduce machine vibration and improve surface finish, but requires tuning additional parameters.

Aktivieren

Zu setzen in der INI-Datei:

```
[TRAJ]
PLANNER_TYPE = 1              # 0=trapezoidal (default), 1=S-curve
MAX_LINEAR_JERK = 1000.0      # Machine units/s^3

[JOINT_n]
MAX_JERK = 1000.0
```

S-curve planning is only active when `PLANNER_TYPE = 1` and `MAX_LINEAR_JERK > 0`.

NOTE If `MAX_LINEAR_JERK` is not specified, it defaults to 1e9 (1 billion), which effectively disables jerk limiting while maintaining S-curve calculations. This produces motion similar to trapezoidal planning but not identical. The maximum allowed value is 1e9 to prevent numerical instability.

Feineinstellung (engl. tuning)

Start with a conservative jerk value and increase gradually:

```
MAX_JERK ≈ 10 to 100 × MAX_ACCELERATION
```

Typical values: 100-100,000 units/s^3 depending on machine rigidity and units (mm values are typically 1000x larger than inch values).

Increase `MAX_LINEAR_JERK` until motion becomes sluggish or following errors increase, then reduce slightly. Test with coordinated moves and arcs.

Values above 1e9 are automatically clamped to 1e9 to avoid numerical issues in the S-curve trajectory calculations.

4.1.6. RTAI

Die Echtzeit-Anwendungsschnittstelle (Real Time Application Interface, RTAI) wird verwendet, um die beste Echtzeitleistung (RT) zu erzielen. Mit dem gepatchten RTAI-Kernel können Sie Anwendungen mit strengen Zeitvorgaben schreiben. RTAI gibt Ihnen die Möglichkeit, Dinge wie die Software-Schritterzeugung durchzuführen, die ein präzises Timing erfordern.

ACPI

Das Advanced Configuration and Power Interface (ACPI) hat viele verschiedene Funktionen, von denen die meisten die RT-Leistung beeinträchtigen (z. B.: Energieverwaltung, CPU-Abschaltung, CPU-Frequenzskalierung usw.). Der LinuxCNC-Kernel (und wahrscheinlich alle RTAI-gepatchten Kernel) hat ACPI deaktiviert. ACPI kümmert sich auch um das Herunterfahren des Systems, nachdem ein Shutdown gestartet wurde, und deshalb müssen Sie möglicherweise den Netzschalter drücken, um Ihren Computer vollständig auszuschalten. Die RTAI-Gruppe hat dies in den letzten Versionen verbessert, so dass sich Ihr LinuxCNC-System vielleicht doch von selbst ausschaltet.

4.1.7. Optionen für Hardware als Schnittstelle zwischen Computer und Maschine

litehm2/rv901t

Litehm2 ist ein Board-agnostischer Port der HostMot2-FPGA-Firmware. Das erste unterstützte Board ist das linsn rv901t, das ursprünglich als LED-Controller-Board gebaut wurde, aber aufgrund der verfügbaren I/Os auch gut als Maschinencontroller geeignet ist. Es bietet etwa 80 5V-gepufferte I/O-Ports und kann zwischen allen Eingängen und allen Ausgängen umschalten. Es kann auch leicht modifiziert werden, um die Ports halb/halb zwischen Eingang und Ausgang aufzuteilen. Der rv901t wird über Gigabit- oder 100-Mbit-Ethernet mit dem Computer verbunden.

Litehm2 basiert auf dem LiteX-Framework, das eine breite Palette von FPGA-Boards unterstützt. Derzeit wird nur das rv901t unterstützt, aber die Unterstützung für weitere Boards ist in der Entwicklung.

Weitere Informationen finden Sie unter https://github.com/sensille/litehm2.

4.2. Latenz-Test

4.2.1. Was bedeutet Latenz?

Die Latenz ist die Zeit, die der PC braucht, um seine reguläre Arbeit zu unterbrechen und auf eine externe Anfrage zu reagieren, z. B. das Ausführen eines der regelmäßigen Echtzeit-Threads von LinuxCNC. Je niedriger die Latenz, desto schneller können Sie die Echtzeit-Threads ausführen, und desto flüssiger wird die Bewegung (und möglicherweise schneller im Fall von Software-Stepping).

Die Latenzzeit ist viel wichtiger als die CPU-Geschwindigkeit. Ein bescheidener Pentium II, der auf Unterbrechungen jedes Mal innerhalb von 10 Mikrosekunden reagiert, kann bessere Ergebnisse liefern als das neueste und schnellste P4-Hyperthreading-Biest.

Die CPU ist nicht der einzige Faktor, der die Latenzzeit bestimmt. Motherboards, Grafikkarten, USB-Anschlüsse und eine Reihe anderer Dinge können die Latenz beeinträchtigen. Der beste Weg, um herauszufinden, womit Sie es zu tun haben, ist die Durchführung des Latenztests.

Die Erzeugung von Schrittimpulsen in der Software hat einen sehr großen Vorteil - sie ist kostenlos. So gut wie jeder PC mit einer parallelen Schnittstelle ist auch in der Lage, die durch Software ausgelösten Schrittimpulse auszugeben.

However, software step pulses also have some disadvantages:

- begrenzte maximale Schrittfrequenz
- Jitter (variierende zeitliche Abstände) in den erzeugten Impulsen
- belastet die CPU

4.2.2. Latenz-Tests

LinuxCNC enthält mehrere Latenz-Tests. Sie alle produzieren gleichwertige Informationen. Das Ausführen dieser Tests wird helfen, festzustellen, ob ein Computer für den Betrieb einer CNC-Maschine geeignet ist.

NOTE Führen Sie LinuxCNC oder StepConf nicht aus, während der Latenztest läuft.

Latenz-Test

The latency test can be run a few different ways.

If you are using PnCconf to configure your machine, you can launch the Latency Test by clicking the "Test Base Period Jitter button' during the 2nd step of the process.

If you are using StepConf to configure your machine, you can launch the Latency Test by clicking the "Test Base Period Jitter button' during the 2nd step of the process.

Um den Test von der Kommandozeile auszuführen, öffnen Sie ein Terminal-Fenster (in Ubuntu, von Anwendungen → Zubehör → Terminal) und führen den folgenden Befehl aus:

```
latency-test
```

Damit wird der Latenztest mit einer Basis-Thread-Periode von 25 µs und einer Servo-Thread-Periode von 1 ms gestartet. Die Periodenzeiten können in der Befehlszeile angegeben werden:

```
latency-test 50000 1000000
```

Damit wird der Latenztest mit einer Basis-Thread-Periode von 50 µs und einer Servo-Thread-Periode von 1 ms gestartet.

Die verfügbaren Optionen können Sie in der Befehlszeile eingeben:

```
latency-test -h
```

Nach dem Starten eines Latenztests sollten Sie so etwas sehen:

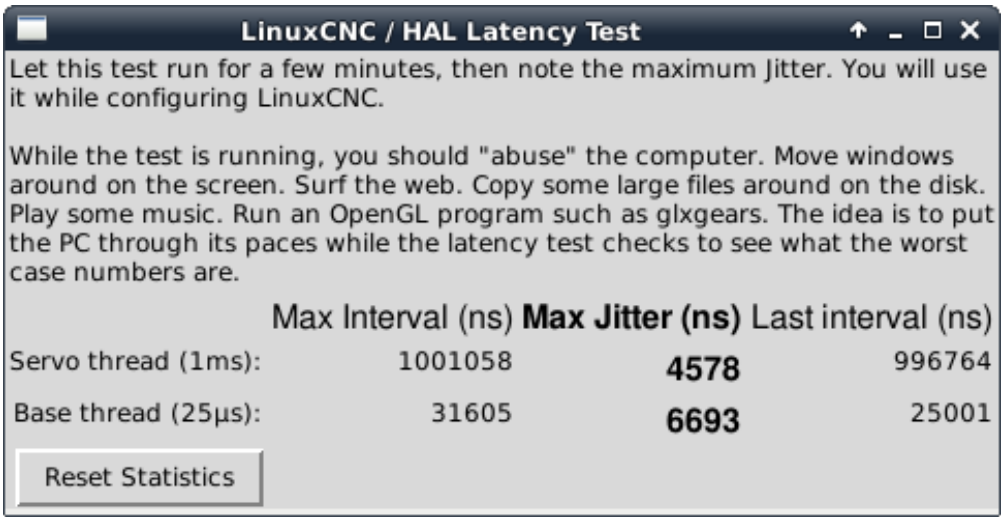


Figure 56. HAL-Latenz-Test

Während der Test läuft, sollten Sie den Computer beschäftigen: Bewegen Sie die Fenster auf dem Bildschirm. Surfen Sie im Internet. Kopieren Sie einige große Dateien auf der Festplatte. Spielen Sie etwas Musik ab. Führen Sie ein OpenGL-Programm wie z. B. glxgears aus. Die Idee ist, den PC auf Herz und Nieren zu prüfen, während der Latenztest den schlimmsten Fall ermittelt.

Die wichtige Zahl für das Software-Stepping ist der "maximale Jitter" des Basis-Threads. Im obigen Beispiel sind das 6693 Nanosekunden (ns), oder 6,693 Mikrosekunden (µs). Notieren Sie diese Zahl und geben Sie sie in StepConf ein, wenn sie angefordert wird.

Im obigen Beispiel lief der Latenztest nur einige Sekunden lang. Sie sollten den Test mindestens mehrere Minuten lang durchführen; manchmal tritt die schlimmste Latenz nicht sehr oft auf oder nur, wenn Sie eine bestimmte Aktion durchführen. Eine Intel-Hauptplatine funktionierte beispielsweise die meiste Zeit

recht gut, aber alle 64 Sekunden hatte sie eine sehr schlechte 300 µs-Latenz. Glücklicherweise war das behebbar, siehe Latenz-Tuning.

So, what do the results mean?

If your Max Jitter number is less than about 20,000 nanoseconds, the computer should give very nice results with software stepping or a dedicated hardware card such as a Mesa *Anything I/O* card.

If the Max Jitter number is between 20,000 and 50,000 nanoseconds, you can still get good results with software stepping, but your maximum step rate might be a little disappointing, especially if you use microstepping or have very fine pitch leadscrews. You can, however, achieve excellent results using a hardware card.

If the Max Jitter number is between 50,000 and 500,000 nanoseconds, you cannot use software stepping. You can, however, achieve acceptable results using a hardware card.

If the Max Jitter number is above 500,000 nanoseconds, you cannot use software stepping or a hardware card with LinuxCNC and achieve acceptable results.

NOTE Wenn Sie hohe Zahlen erhalten, gibt es möglicherweise Möglichkeiten, sie zu verbessern. Ein anderer PC hatte eine sehr schlechte Latenz (mehrere Millionen Nanosekunden, d.h. Millisekunden) bei der Verwendung des Onboard-Videos. Aber eine €5 gebrauchte Grafikkarte löste das Problem. LinuxCNC benötigt keine hochmoderne Hardware.

Weitere Informationen zum Stepper-Tuning finden Sie im Kapitel Stepper Tuning.

TIP Zusätzliche Kommandozeilen-Tools sind für die Untersuchung der Latenz verfügbar wenn LinuxCNC nicht läuft.

Latency Plot

latency-plot erstellt ein Streifendiagramm für einen Basis- und einen Servo-Thread. Es kann nützlich sein, um Spitzen in der Latenz zu sehen, wenn andere Anwendungen gestartet oder verwendet werden. Verwendung:

```
latency-plot --help

Usage:
      latency-plot --help | -?
      latency-plot --hal [Options]

Optionen:
      --base ns  (base  thread interval in nanoseconds, default:   25000)
      --servo ns (servo thread interval in nanoseconds, default: 1000000)
      --time ms  (report interval in milliseconds, default: 1000)
      --relative (relative clock time (default))
      --actual   (actual clock time)
```

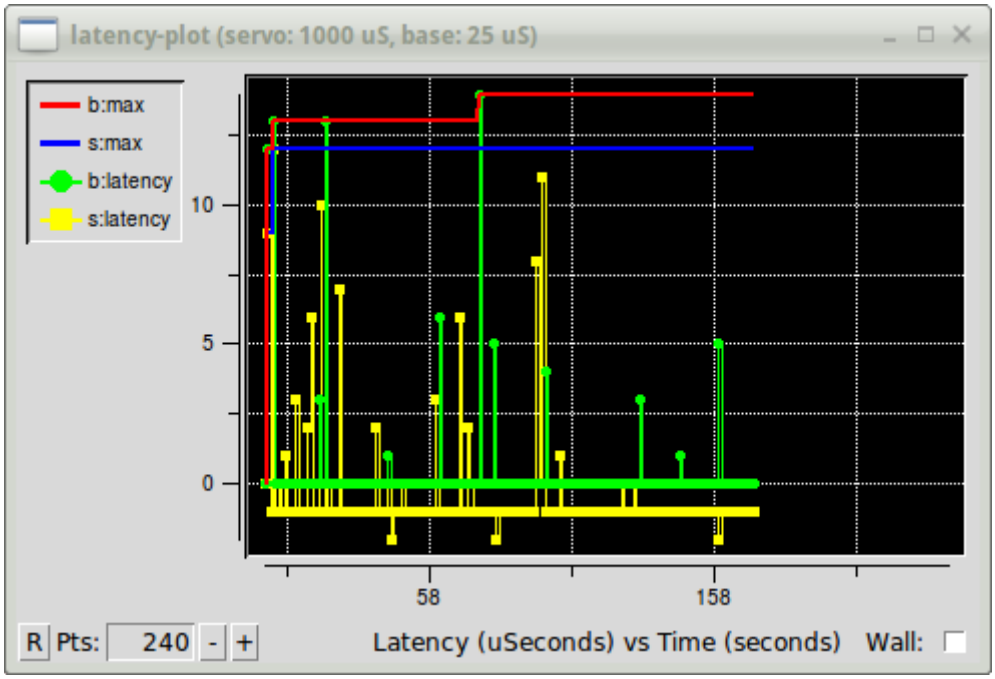


Figure 57. `latency-plot`-Fenster

Latenz-Histogramm

Die Anwendung latency-histogram zeigt ein Histogramm der Latenz (Jitter) für einen Basis- und einen Servo-Thread an.

```
Usage:
   latency-histogram --help | -?
   latency-histogram [Options]

Optionen:
  --base ns (Basisgewindeintervall in Nanosekunden, Voreinstellung: 25000, min: 5000)
  --servo ns (Servo-Thread-Intervall in Nanosekunden, Voreinstellung: 1000000,
Mindestwert: 25000)
  --bbinsize ns (Basis-Bin-Größe in Nanosekunden, Voreinstellung: 100
  --sbinsize ns (Servo-Bin-Größe in Nanosekunden, Voreinstellung: 100)
  --bbins n (Basis-Bins, Voreinstellung: 200
  --sbins n (Servo-Bins, Voreinstellung: 200
  --logscale 0|1 (logarithmische Skala der y-Achse, Voreinstellung: 1)
  --text note (zusätzlicher Hinweis, Voreinstellung: "" )
  --show (zeigt die Anzahl der nicht angezeigten Bins)
  --nobase (nur Servo-Thread)
  --verbose (Fortschritt und Fehlersuche)
  --nox (keine Benutzeroberfläche, Anzeige von elapsed,min,max,sdev für jeden Thread)
```

NOTE

Während die Latenz bestimmt wird, sollten LinuxCNC und HAL nicht ausgeführt werden, stoppen Sie mit `halrun -U`. Eine große Anzahl von Bins und/oder kleine Binsizes verlangsamen die Aktualisierung. Für einen einzelnen Thread geben Sie **--nobase** (und Optionen für den Servo-Thread) an. Gemessene Latenzen außerhalb des +/- Bereichs werden mit speziellen Endbalken angezeigt. Verwenden Sie **--show**, um die Zählung für den Off-Chart [pos|neg] bin anzuzeigen.

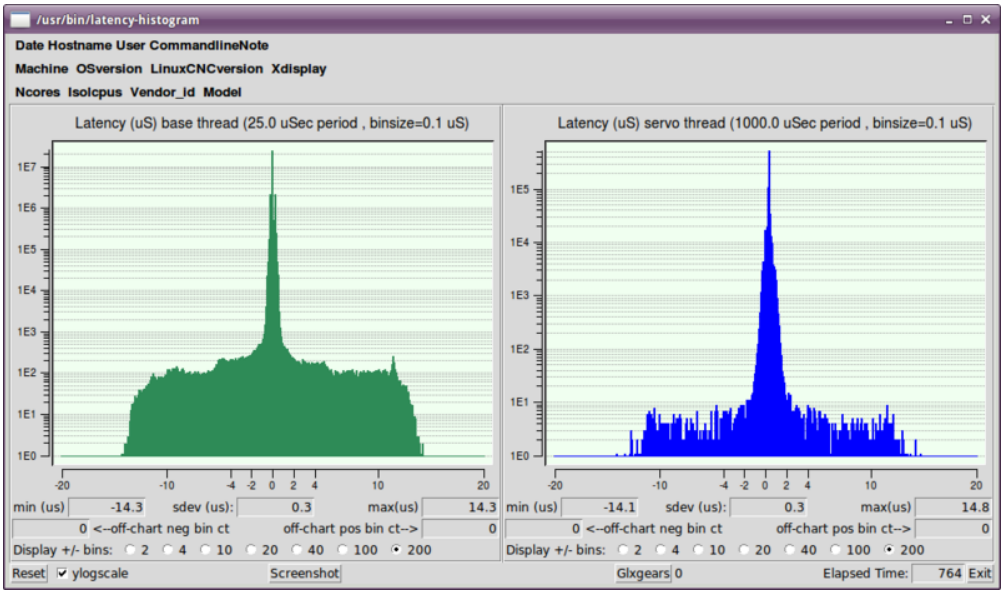


Figure 58. Latenz-Histogramm-Fenster

Realtime status bar

The `latency-test` and `latency-histogram` windows show a status bar along the bottom with three fields. A field turns red when it indicates a condition that makes the measured latency unrepresentative of a properly configured realtime system.

Echtzeit

The realtime type that rtapi is running, as reported by `realtime verify`, for example `Preempt RT`, `RTAI` or `Xenomai`. It reads `No realtime` (red) when `rtapi_app` cannot obtain realtime scheduling, either because the running kernel is not a realtime kernel (see Kernel and Version requirements) or because `rtapi_app` lacks the required privileges on a run-in-place build (see Realtime, run `sudo make setuid` or `sudo make setcap`). In this state the figures reflect ordinary, non-realtime scheduling and will be far worse than a realtime kernel delivers. For testing only, realtime can be forced with the `LINUXCNC_FORCE_REALTIME=1` environment variable.

CPU governor

The active CPU frequency governor. It is shown in red when it is not `performance`, because frequency scaling lets the CPU drop to a lower clock speed and adds latency spikes. See CPU frequency governor.

isolcpus

Whether any CPUs are isolated for the exclusive use of LinuxCNC (`isolcpus=...`) or not (`no_isolcpus`). Isolating a CPU keeps other tasks off it and improves latency; see Tuning Preempt-RT for latency.

4.2.3. Latenz-Tuning

LinuxCNC kann auf vielen verschiedenen Hardware-Plattformen und mit vielen verschiedenen Echtzeit-Kernel laufen, und sie alle können von Tuning für eine optimale Latenz profitieren.

Ein primäres Ziel bei der Abstimmung des Systems für LinuxCNC ist es, eine CPU für die ausschließliche

Verwendung von LinuxCNCs Echtzeit-Tasks zu reservieren, so dass andere Tasks (sowohl Benutzerprogramme als auch Kernel-Threads) den Zugriff von LinuxCNC auf diese CPU nicht stören.

Wenn bestimmte Tuning-Optionen werden geglaubt, um universell hilfreich LinuxCNC tut diese Abstimmung automatisch beim Start, aber viele Tuning-Optionen sind maschinenspezifisch und kann nicht automatisch durchgeführt werden. Die Person, die LinuxCNC installiert, muss experimentell die optimale Abstimmung für ihr System zu bestimmen.

Optimieren des BIOS für die Latenz

PC-BIOSe unterscheiden sich stark in ihrem Latenzverhalten.

Das Tuning des BIOS ist mühsam, da Sie den Computer neu starten, eine kleine Änderung im BIOS vornehmen, Linux booten und den Latenztest (möglicherweise für eine lange Zeit) ausführen müssen, um zu sehen, welche Auswirkungen Ihre BIOS-Änderung hatte. Wiederholen Sie dann alle anderen BIOS-Einstellungen, die Sie ausprobieren möchten.

Da BIOS-Systeme alle unterschiedlich und nicht standardmäßig sind, ist die Bereitstellung einer detaillierten BIOS-Tuning-Anleitung nicht praktikabel. Im Allgemeinen sind einige Dinge, die Sie im BIOS versuchen sollten:

- Deaktivieren Sie ACPI, APM und alle anderen Energiesparfunktionen. Dies schließt alles ein, was mit Stromsparen, Suspendieren, CPU-Ruhezuständen, CPU-Frequenzskalierung usw. zu tun hat.
- Deaktivieren Sie den "Turbo"-Modus der CPU.
- Deaktivieren Sie CPU-Hyperthreading.
- Deaktivieren Sie den System Management Interrupt (SMI) (oder kontrollieren Sie ihn anderweitig).
- Deaktivieren Sie alle Hardware, die Sie nicht verwenden wollen.

CPU frequency governor

Frequency scaling lets the CPU lower its clock speed to save power, which adds latency while the CPU ramps back up to service a realtime thread. For the best latency, set the governor to `performance` on every CPU:

```
echo performance | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

This setting does not survive a reboot. To make it permanent, apply it at boot with a small systemd service, or with a CPU frequency management tool such as `cpupower` (Debian/Ubuntu package `linux-cpupower`) or `tuned`.

Optimieren von Preempt-RT für Latenz

Der Preempt-RT-Kernel kann vom Tuning profitieren, um die beste Latenz für LinuxCNC zu erreichen. Die Abstimmung kann über die Kernel-Befehlszeile, sysctl, und über Dateien in `/proc` und `/sys` erfolgen.

Einige Tuning-Parameter, die es zu beachten gilt:

Kernel-Befehlszeile

Einzelheiten hier: https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt

- `isolcpus`: Verhindert, dass die meisten Nicht-LinuxCNC-Prozesse diese CPUs benutzen, so dass mehr CPU-Zeit für LinuxCNC zur Verfügung steht.
- irqaffinity`: Wählen Sie, welche CPUs Interrupts bedienen, so dass die CPUs, die für LinuxCNC Echtzeit reserviert sind, diese Aufgabe nicht übernehmen müssen.
- `rcu_nocbs`: Verhindert die Ausführung von RCU-Callbacks auf diesen CPUs.
- `rcu_nocb_poll`: Suche nach RCU-Callbacks statt Sleep/Wake zu verwenden.
- `nohz_full`: Deaktiviert den Takt auf diesen CPUs.

Sysctl

Einzelheiten hier: https://www.kernel.org/doc/html/latest/scheduler/sched-rt-group.html

- `sysctl.kernel.sched_rt_runtime_us`: Setzen Sie diesen Wert auf -1, um die Begrenzung der Zeit aufzuheben, die Echtzeit-Tasks verwenden dürfen.

4.3. Stepper-Abstimmung

4.3.1. Das Beste aus Software Stepping herausholen

Die Erzeugung von Schrittimpulsen in der Software hat einen sehr großen Vorteil - sie ist kostenlos. Nahezu jeder PC verfügt über eine parallele Schnittstelle, die in der Lage ist, die von der Software erzeugten Schrittimpulse auszugeben. Die Software-Schrittimpulse haben jedoch auch einige Nachteile:

- begrenzte maximale Schrittfrequenz
- Jitter (variierende zeitliche Abstände) in den erzeugten Impulsen
- belastet die CPU

In diesem Kapitel finden Sie einige Schritte, die Ihnen dabei helfen können, die besten Ergebnisse aus softwaregenerierten Schritten zu erzielen.

Führen Sie einen Latenztest durch

Die CPU ist nicht der einzige Faktor, der die Latenzzeit bestimmt. Motherboards, Grafikkarten, USB-Anschlüsse und viele andere Dinge können die Latenz beeinträchtigen. Der beste Weg, um zu wissen, was man von einem PC erwarten kann, ist, die RT-Latenztests durchzuführen.

Führen Sie den Latenztest wie im Kapitel Latenz-Test beschrieben durch.

Während der Test läuft, sollten Sie den Computer "missbrauchen". Bewegen Sie die Fenster auf dem Bildschirm. Surfen Sie im Internet. Kopieren Sie einige große Dateien auf der Festplatte. Spielen Sie etwas Musik ab. Führen Sie ein OpenGL-Programm wie z. B. glxgears aus. Die Idee ist, den PC auf Herz und Nieren zu prüfen, während der Latenztest den schlimmsten Fall ermittelt.

Die letzte Zahl in der Spalte "Max Jitter" ist die wichtigste. Schreiben Sie sie auf - Sie werden sie später brauchen. Sie enthält die schlechteste Latenzmessung während des gesamten Testlaufs. Im obigen Beispiel sind das 6693 Nanosekunden, also 6,69 Mikrosekunden, was hervorragend ist. Allerdings lief das Beispiel nur einige Sekunden lang (es wird jede Sekunde eine Zeile gedruckt). Sie sollten den Test mindestens mehrere Minuten lang durchführen; manchmal tritt die Latenz im schlimmsten Fall nicht sehr oft auf oder nur, wenn Sie eine bestimmte Aktion durchführen. Ich hatte eine Intel-Hauptplatine, welche die meiste Zeit recht gut funktionierte, aber alle 64 Sekunden eine sehr schlechte Latenz von 300 µs hatte. Glücklicherweise ist das behebbar, siehe Fixing SMI issues on the LinuxCNC Wiki

Was bedeuten also die Ergebnisse? Wenn Ihre "Max Jitter"-Zahl weniger als 15-20 Mikrosekunden (15000-20000 Nanosekunden) beträgt, sollte der Computer mit Software-Stepping sehr gute Ergebnisse liefern. Wenn die maximale Latenzzeit eher bei 30-50 Mikrosekunden liegt, können Sie immer noch gute Ergebnisse erzielen, aber Ihre maximale Schrittrate könnte etwas enttäuschend sein, insbesondere wenn Sie Mikroschrittverfahren verwenden oder sehr feine Spindelsteigungen haben. Wenn die Zahlen 100 µs oder mehr (100.000 Nanosekunden) betragen, ist der PC kein guter Kandidat für Software-Stepping. Zahlen über 1 Millisekunde (1.000.000 Nanosekunden) bedeuten, dass der PC ist kein guter Kandidat für LinuxCNC, unabhängig davon, ob Sie Software-Stepping verwenden oder nicht.

Beachten Sie, dass, wenn Sie hohe Zahlen erhalten, es Möglichkeiten geben kann, sie zu verbessern. Zum Beispiel hatte ein PC eine sehr schlechte Latenz (mehrere Millisekunden), wenn er das Onboard-Video verwendete. Aber eine $ 5 gebrauchte Grafikkarte löste das Problem - LinuxCNC benötigt keine modernste Hardware.

Finden Sie heraus, was Ihre Antriebe erwarten

Verschiedene Marken von Schrittmotorantrieben haben unterschiedliche Zeitanforderungen an ihre Schritt- und Richtungseingänge. Sie müssen also das Datenblatt mit den technischen Daten Ihres Antriebs heraussuchen (oder danach googeln).

Aus dem Handbuch des Gecko G202:

```
Schrittfrequenz: 0 bis 200 kHz
Schrittimpuls "0" Zeit: 0.5µs min (Schritt bei fallender Flanke)
Schrittimpuls "1" Zeit: 4.5 µs min
Richtung Setup: 1 µs min (20 µs min Haltezeit nach Schrittflanke)
```

Aus dem Gecko G203V Handbuch:

```
Schrittfrequenz: 0 bis 333 kHz
Schrittimpuls "0" Zeit: 2.0 µs min (Schritt bei steigender Flanke)
Schrittimpuls "1" Zeit: 1.0 µs min

Direction Setup:
    200 ns (0.2 µs) before step pulse rising edge
    200 ns (0.2 µs) hold after step pulse rising edge
```

Aus dem Xylotex-Datenblatt:

```
Minimale DIR-Setup-Zeit vor steigender Flanke des STEP-Impulses 200 ns Minimale
```

```
DIR-Haltezeit nach steigender Flanke des STEP-Pulses 200 ns
Minimale STEP-Impuls-Hochzeit 2,0 µs
Minimale STEP-Impuls-Low-Zeit 1,0 µs
Schritt erfolgt bei steigender Flanke
```

Wenn Sie die Zahlen gefunden haben, notieren Sie sie ebenfalls - Sie brauchen sie im nächsten Schritt.

Wählen Sie Ihren BASE_PERIOD

BASE_PERIOD ist der *Herzschlag* Ihres LinuxCNC Computers. Jede Periode, die Software-Schritt-Generator entscheidet, ob es Zeit für einen weiteren Schritt Impuls ist. Eine kürzere Periode ermöglicht es Ihnen, mehr Impulse pro Sekunde, innerhalb von Grenzen zu erzeugen. Aber wenn Sie zu kurz gehen, wird Ihr Computer so viel Zeit damit verbringen, Schrittimpulse zu erzeugen, dass alles andere zu einem Kriechgang verlangsamen wird, oder vielleicht sogar sperren. Die Latenzzeit und die Anforderungen an die Schrittmotorsteuerung beeinflussen die kürzeste Periode, die Sie verwenden können, wie wir gleich sehen werden.

Schauen wir uns zuerst das Gecko-Beispiel an. Der G202 kann Schrittimpulse verarbeiten, die 0,5 µs lang auf low und 4,5 µs lang auf high gehen. Der Richtungs-Pin muss 1 µs vor der fallenden Flanke stabil sein und nach der fallenden Flanke 20 µs lang stabil bleiben. Die längste Zeitanforderung ist die Haltezeit von 20 µs. Ein einfacher Ansatz wäre, die Periode auf 20 µs zu setzen. Das bedeutet, dass alle Änderungen an den STEP- und DIR-Leitungen durch 20 µs getrennt sind. Alles ist gut, oder?

Falsch! Wenn es NULL Latenz gäbe, dann wären alle Kanten durch 20 µs getrennt, und alles wäre in Ordnung. Aber alle Computer haben eine gewisse Latenz, d.h. mit Verzögerung. Wenn der Computer eine Latenz von 11 µs hat, bedeutet das, dass die Software manchmal 11 µs später läuft, als sie eigentlich sollte. Wenn ein Durchlauf der Software 11 µs zu spät ist und der nächste pünktlich erfolgt, beträgt die Verzögerung vom ersten zum zweiten Durchlauf nur 9 µs. Wenn der erste Durchlauf einen Schrittimpuls erzeugte und der zweite das Richtungsbit änderte, haben Sie gerade die G202-Haltezeitanforderung von 20 µs verletzt. Das bedeutet, dass Ihr Antrieb möglicherweise einen Schritt in die falsche Richtung gemacht hat, und Ihr Teil hat die falsche Größe.

Das wirklich Unangenehme an diesem Problem ist, dass es sehr selten auftreten kann. Im schlimmsten Fall treten Latenzen nur ein paar Mal pro Minute auf, und die Wahrscheinlichkeit, dass eine schlechte Latenz genau dann auftritt, wenn der Motor die Richtung ändert, ist gering. So kommt es zu sehr seltenen Fehlern, die hin und wieder ein Werkstück ruinieren und eine Fehlerbehebung unmöglich machen.

Der einfachste Weg, dieses Problem zu vermeiden, besteht darin, eine BASE_PERIOD zu wählen, die der Summe aus der längsten Zeitanforderung Ihres Laufwerks und der schlimmsten Latenz Ihres Computers entspricht. Wenn Sie einen Gecko mit einer Haltezeitanforderung von 20 µs betreiben und Ihr Latenztest eine maximale Latenz von 11 µs ergab, dann können Sie, wenn Sie die BASE_PERIOD auf 20+11 = 31 µs (31000 Nanosekunden in der INI-Datei) setzen, die Timing-Anforderungen des Laufwerks garantiert erfüllen.

Aber es gibt einen Kompromiss. Für einen Stufenimpuls sind mindestens zwei Perioden erforderlich. Eine, um den Impuls zu starten, und eine, um ihn zu beenden. Da die Periode 31 µs beträgt, dauert es 2x31 = 62 µs, um einen Schrittimpuls zu erzeugen. Das bedeutet, dass die maximale Schrittfrequenz nur 16.129 Schritte pro Sekunde beträgt. Das ist nicht so gut. (Aber geben Sie noch nicht auf, wir müssen im

nächsten Abschnitt noch einige Optimierungen vornehmen.)

Beim Xylotex sind die Setup- und Haltezeiten mit jeweils 200 ns (0,2 µs) sehr kurz. Die längste Zeit ist die 2-µs-High-Zeit. Wenn Sie eine Latenzzeit von 11 µs haben, dann können Sie die BASE_PERIOD auf 11+2=13 µs einstellen. Die lange Haltezeit von 20 µs entfällt, was sehr hilfreich ist! Bei einer Periode von 13 µs dauert ein kompletter Schritt 2x13 = 26 µs, und die maximale Schrittrate beträgt 38.461 Schritte pro Sekunde!

Aber Sie können noch nicht mit dem Feiern anfangen. Beachten Sie, dass 13 µs ein sehr kurzer Zeitraum ist. Wenn Sie versuchen, die Schritt-Generator alle 13 µs laufen, könnte es nicht genug Zeit übrig, um etwas anderes laufen, und Ihr Computer wird einfrieren. Wenn Sie für Zeiträume von weniger als 25 µs anstreben, sollten Sie bei 25 µs oder mehr beginnen, führen Sie LinuxCNC, und sehen, wie die Dinge reagieren. Wenn alles gut ist, können Sie allmählich den Zeitraum zu verringern. Wenn der Mauszeiger beginnt immer träge, und alles andere auf dem PC verlangsamt, ist Ihr Zeitraum ein wenig zu kurz. Gehen Sie zurück zu dem vorherigen Wert, der den Computer reibungslos laufen lässt.

Nehmen wir an, Sie haben mit 25 µs begonnen und versuchen, auf 13 µs zu kommen, aber Sie stellen fest, dass 16 µs die Grenze sind - bei weniger reagiert der Computer nicht sehr gut. Sie verwenden also 16 µs. Bei einer Periode von 16 µs und einer Latenzzeit von 11 µs ist die kürzeste Ausgabezeit 16-11 = 5 µs. Das Laufwerk braucht nur 2 µs, also haben Sie etwas Spielraum. Ein gewisser Spielraum ist gut, denn Sie wollen keine Schritte verlieren, weil Sie das Timing zu knapp gewählt haben.

Was ist die maximale Schrittgeschwindigkeit? Denken Sie daran, zwei Perioden für einen Schritt. Sie haben sich auf 16 µs für die Periode geeinigt, also dauert ein Schritt 32 µs. Das ergibt nicht schlechte 31.250 Schritte pro Sekunde.

Verwenden Sie steplen, stepspace, dirsetup und/oder dirhold

Im letzten Abschnitt haben wir das Xylotex-Laufwerk auf eine Zeitspanne von 16 µs und eine maximale Geschwindigkeit von 31.250 Schritten pro Sekunde gebracht. Aber der Gecko blieb bei 31 µs und nicht ganz so schönen 16.129 Schritten pro Sekunde stecken. Das Xylotex-Beispiel ist so gut, wie wir es machen können. Aber der Gecko kann noch verbessert werden.

Das Problem mit dem G202 ist die erforderliche Haltezeit von 20 µs. Das plus die 11 µs Latenzzeit ist das, was uns zwingt, eine langsame 31 µs Periode zu verwenden. Aber die LinuxCNC Software-Schritt-Generator hat einige Parameter, mit denen Sie die verschiedenen Zeit von einer Periode auf mehrere zu erhöhen. Zum Beispiel, wenn steplen von 1 auf 2 geändert wird, dann wird es zwei Perioden zwischen dem Beginn und dem Ende des Schrittimpulses sein. Wenn dirhold von 1 auf 3 geändert wird, liegen mindestens drei Perioden zwischen dem Schrittimpuls und einem Wechsel des Richtungspins.

Wenn wir dirhold verwenden können, um die Anforderung von 20 µs Haltezeit zu erfüllen, dann ist die nächstlängere Zeit die 4,5 µs "high time". Addiert man die Latenzzeit von 11 µs zu der "high-time" von 4,5 µs, so erhält man eine Mindestzeit von 15,5 µs. Wenn Sie 15,5 µs ausprobieren, stellen Sie fest, dass der Computer zu träge ist, also entscheiden Sie sich für 16 µs. Wenn wir dirhold auf 1 belassen (die Voreinstellung), dann ist die Mindestzeit zwischen Schritt und Richtung die 16 µs Periode minus die 11 µs Latenzzeit = 5 µs, was nicht ausreicht. Wir brauchen weitere 15 µs. Da die Periode 16 µs beträgt, brauchen wir eine weitere Periode. Also ändern wir dirhold von 1 auf 2. Jetzt beträgt die Mindestzeit vom Ende des Schrittimpulses bis zum Richtungswechsel 5+16=21 µs, und wir müssen uns keine Sorgen

mehr machen, dass der Gecko wegen der Latenz die falsche Richtung einschlägt.

Wenn der Computer eine Latenzzeit von 11 µs hat, dann stellt eine Kombination aus einer Basisperiode von 16 µs und einem Dirhold-Wert von 2 sicher, dass wir die Timing-Anforderungen des Gecko immer erfüllen. Bei normalem Steppen (ohne Richtungswechsel) hat der erhöhte dirhold-Wert keine Auswirkung. Es werden zwei Perioden von insgesamt 32 µs für jeden Schritt benötigt, und wir haben die gleiche Schrittrate von 31.250 Schritten pro Sekunde wie beim Xylotex.

Die in diesem Beispiel verwendete Latenzzeit von 11 µs ist sehr gut. Wenn Sie diese Beispiele mit einer größeren Latenzzeit, z. B. 20 oder 25 µs, durcharbeiten, wird die Spitzenschrittrate sowohl für den Xylotex als auch für den Gecko niedriger sein. Es gelten jedoch dieselben Formeln für die Berechnung der optimalen BASE_PERIOD und für die Anpassung der Dirhold- oder anderer Schrittgeneratorparameter.

Nicht raten!

Um ein schnelles UND zuverlässiges softwarebasiertes Steppersystem zu erhalten, können Sie die Perioden und andere Konfigurationsparameter nicht einfach erraten. Sie müssen auf Ihrem Computer Messungen vornehmen und die Berechnungen durchführen, um sicherzustellen, dass Ihre Antriebe die benötigten Signale erhalten.

Um die Berechnungen zu vereinfachen, habe ich eine Open Office-Tabelle Step Timing Calculator erstellt. Sie geben das Ergebnis Ihres Latenztests und die Anforderungen an die Schrittmotorsteuerung ein, und die Kalkulationstabelle berechnet die optimale BASE_PERIOD. Anschließend testen Sie die Periode, um sicherzustellen, dass sie Ihren PC nicht verlangsamt oder blockiert. Schließlich geben Sie die tatsächliche Periode ein, und die Kalkulationstabelle zeigt Ihnen die Stepgen-Parametereinstellungen an, die erforderlich sind, um die Timing-Anforderungen Ihres Antriebs zu erfüllen. Es berechnet auch die maximale Schrittrate, die Sie erzeugen können.

Ich habe der Tabelle ein paar Dinge hinzugefügt, um die maximale Geschwindigkeit und die elektrischen Berechnungen der Stepper zu berechnen.

4.4.INI-Konfiguration

4.4.1.Die INI-Datei-Komponenten

Eine typische INI-Datei hat ein recht einfaches Layout, das Folgendes umfasst;

- Kommentare
- Abschnitte
- Variablen

Jedes dieser Elemente wird in einzelnen Zeilen getrennt. Jedes Zeilenende oder Zeilenumbruchzeichen erzeugt ein neues Element.

Kommentare

A comment line is started with a ; or a # mark. When the INI reader sees either of these marks on a line, the rest of the line is ignored by the software. Comments can be used to describe what an INI element will do.

```
; Dies ist die Konfigurationsdatei meiner Fräsmaschine
# Ich habe sie am 12. Januar 2012 eingerichtet.
```

Kommentare können auch zum *Ausschalten* einer Variable verwendet werden. Das macht es einfacher, zwischen verschiedenen Variablen zu wählen.

```
DISPLAY = axis
# DISPLAY = touchy
```

In dieser Liste wird die Variable DISPLAY auf axis gesetzt, weil die andere auskommentiert ist. Wenn jemand unvorsichtigerweise eine Liste wie diese bearbeitet und zwei der Zeilen unkommentiert lässt, wird die zuerst gefundene Zeile verwendet.

Note that inside a variable's value, the "#" and ";" characters are part of the value:

```
# Below does not result in INCORRECT=value
# because comments are not interpreted as comments in values
INCORRECT = value # and this is not a comment

# Correct comment
# hash char # is a comment om this line
CORRECT = value
```

Abschnitte

Related parts of an INI file are separated into sections. A section name is enclosed in brackets like this: `[THIS_SECTION]`. The order of sections is unimportant. Sections begin at the section name and end at the next section name. Section identifiers are case senstive and can only contain letters A-Z, a-z, digits 0-9 and underscore (_). Additionally, section identifiers cannot start with a digit.

Die folgenden Abschnitte werden von LinuxCNC verwendet:

- `[EMC]` allgemeine Informationen
- `[DISPLAY]` Einstellungen im Zusammenhang mit der grafischen Benutzeroberfläche
- `[FILTER]` Einstellungen für Eingaben-Filterprogramme
- `[RS274NGC]` vom G-Code-Interpreter verwendete Einstellungen
- `[EMCMOT]` Einstellungen, die von der Echtzeit-Bewegungssteuerung verwendet werden
- `[TASK]` vom Task-Controller verwendete Einstellungen
- `[HAL]` gibt HAL-Dateien an
- `[HALUI]` Von HALUI verwendete MDI-Befehle

- `[APPLICATIONS]` Andere Anwendungen, die von LinuxCNC gestartet werden sollen
- `[TRAJ]` zusätzliche Einstellungen, die von der Echtzeit-Bewegungssteuerung verwendet werden
- `[JOINT_n]` einzelne Gelenkvariablen
- `[AXIS_l]` einzelne Achsenvariablen
- `[KINS]` Variablen für die Kinematik
- `[EMCIO]` vom E/A-Controller verwendete Einstellungen

Variablen

A variable line is made up of a variable name, an equals sign (=), and a value. A variable identifier is case sensitive and may only consist of letters A-Z, a-z, digits 0-9 and underscore (_). Additionally, variable identifiers cannot start with a digit. White space at the beginning of the line and after the variable identifier, up to the equals sign, is ignored.

Beispiel für eine Variable

```
MACHINE = My Machine
```

A variable line may be extended to multiple lines with a terminal backslash (\) character. There may be white space following the trailing backslash character, but this is strongly discouraged.

Abschnittsbezeichnungen dürfen nicht auf mehrere Zeilen ausgedehnt werden.

Beispiel für Variable mit Zeilenerweiterung

```
APP = sim_pin \
ini.0.max_acceleration \
ini.1.max_acceleration \
ini.2.max_acceleration \
ini.0.max_velocity \
ini.1.max_velocity \
ini.2.max_velocity
```

A specific variable in a specific sections is often denoted in the documentation as `[SECTION]VARIABLE`. This specification mirrors the same way they are specified in HAL files for expansion.

Variable values may embed special characters in literal or escaped forms. Single or double quotes are not treated as special and are a literal part of the value. Values also support all common escape formats and full Unicode:

- control: `\[abfnrtv]`
- octal: `\[0-2][0-7]{0,2}`
- hex: `\x[0-9a-fA-F]{2}`
- UTF-16: `\u[0-9a-fA-F]{4}`
- UTF-32: `\U[0-9a-fA-F]{8}`

The resulting value is always converted into UTF-8 and checked for validity. It is not allowed to embed

NUL characters either literally or by using an escape.

Leading and trailing white space is normally removed from a value. You can add leading and trailing space in a value by using an escaped value for space (`\x20`) as first or last character. Spaces inside a value are automatically part of the value. Tabs and newlines can be added using `\t` and `\n`.

Value Escape Example

```
STRING = Hello World!

# The following would become: Hello World!
STRING = Hello\
 \
World\
!

SMILE = \370\237\230\200 = 
SMILE = \xf0\x9f\x98\x80 = 
SMILE = \ud83d\ude00 = 
SMILE = \U0001f600 = 
```

Variables' value can have types associated when they are read by LinuxCNC. The types are enforced by the INI file reader when the appropriate function calls are performed. All values may always be read as *string*. Conversion into other types has restrictions. Possible types are:

- *string* - the value is taken as-is
- *boolean* - only boolean words are accepted
- *signed integer* - whole numbers in decimal, hexadecimal, octal or binary bases
- *unsigned integer* - whole numbers in decimal, hexadecimal, octal or binary bases
- *real* - floating point values (always using decimal point)
- *enumeration* - a restricted set of keys to represent a value or function

Signed and unsigned integers are read and converted as 64-bit numbers when marked as `s64` and `u64`. Plain old 32-bit integers are marked `int` and are always signed. The default number base is decimal. Alternative bases may be specified using a prefix. The complete list of valid integer numbers:

- `[0-9]+` (decimal)
- `0x[0-9A-Fa-f]+` (hexadecimal)
- `0o[0-7]+` (octal)
- `0b[01]+` (binary)

Signed integers may be preceded by a plus (+) or minus (-) sign, regardless number base. Unsigned integers will generate a warning if they are preceded by a minus (-) sign upon conversion.

Boolean values are case insensitive and allow the following words:

- true/enabled - `TRUE`, `YES`, `ON` or `1`
- false/disabled - `FALSE`, `NO`, `OFF` or `0`

Enumerations are a set of keywords defined by LinuxCNC and interpreted to mean a setting, value or functionality. The exact values are declared in the individual variable description and the variables are marked with `enum`. Most enumerations are interpreted without case. It is noted in the variable description if case matters.

Some LinuxCNC variables are special in that their format is interpreted as a multi-valued entry. These variables have an appropriate description below of the format expected.

Variables that are used by LinuxCNC must always use the section names and variable names as shown. Any custom or private variables and sections are up to the user.

Benutzerdefinierte Abschnitte und Variablen

Die meisten Beispielkonfigurationen verwenden benutzerdefinierte Abschnitte und Variablen, um alle Einstellungen an einem Ort zu bündeln.

Um eine benutzerdefinierte Variable zu einem bestehenden LinuxCNC-Abschnitt hinzuzufügen, fügen Sie die Variable einfach in diesen Abschnitt ein.

Beispiel für eine benutzerdefinierte Variable, die der Variablen "TYPE" den Wert "LINEAR" und der Variablen "SCALE" den Wert "16000" zuweist.

```
[JOINT_0]
TYPE = LINEAR
# ...
SCALE = 16000
```

Um einen benutzerdefinierten Abschnitt mit eigenen Variablen einzuführen, fügen Sie den Abschnitt und die Variablen in die INI-Datei ein.

Beispiel für einen benutzerdefinierten Abschnitt

```
[PROBE]
Z_FEEDRATE = 50
Z_OFFSET = 12
Z_SAFE_DISTANCE = -10
```

Um die benutzerdefinierten Variablen in Ihrer HAL-Datei zu verwenden, setzen Sie den Abschnitt und den Variablennamen an die Stelle des Wertes.

HAL Beispiel

```
setp offset.1.offset [PROBE]Z_OFFSET
setp stepgen.0.position-scale [JOINT_0]SCALE
```

NOTE Der in der Variablen gespeicherte Wert muss mit dem vom Komponentenpin angegebenen Typ übereinstimmen.

Für benutzerdefinierten Variablen im G-Code verwenden Sie die globale Variablensyntax `#<_ini[section]variable>`. Das folgende Beispiel zeigt eine einfache Z-Achsen-Antastroutine für eine

Oberfräse oder ein Fräswerk unter Verwendung einer Tastplatte.

Please note that G-code embedded ini variables are converted to upper case before they are searched in the INI file. The `#<_ini[section]variable>` should be called [SECTION]VARIABLE in the INI file.

G-Code Beispiel

```
G91
G38.2 Z#<_ini[probe]z_safe_distance> F#<_ini[probe]z_feedrate>
G90
G1 Z#5063
G10 L20 P0 Z#<_ini[probe]z_offset>
```

Include-Dateien

An INI file may include the contents of another file by using a `#INCLUDE` directive. The `#INCLUDE` directive must be in upper case, start in the first column of the line and have at least one space after it.

#INCLUDE Format

```
#INCLUDE filename
```

Der Dateiname kann wie folgt angegeben werden:

- eine Datei in demselben Verzeichnis wie die INI-Datei
- eine Datei, die sich relativ zum Arbeitsverzeichnis befindet
- ein absoluter Dateiname (beginnt mit einem /)
- a user-home-relative file name (starts with a ~/)

Multiple `#INCLUDE` directives are supported.

#INCLUDE Beispiele

```
#INCLUDE joint_0.inc
#INCLUDE ../parallel/joint_1.inc
#INCLUDE below/joint_2.inc
#INCLUDE /home/myusername/myincludes/display.inc
#INCLUDE ~/linuxcnc/myincludes/rs274ngc.inc
```

The `#INCLUDE` directives are supported up to 16 levels — an included file may include additional files up to 16 levels deep. Recursive inclusion of files is detected and flagged as an error. The recommended file extension is *.inc*. Do *not* use a file extension of *.ini* for included files.

4.4.2.INI-Datei Abschnitte

Variables in each section have an associated type as denoted in parentheses after the variable.

[EMC] Abschnitt

- `VERSION = 1.1` - (string) The format version of this configuration. Any value other than 1.1 will cause the configuration checker to run and try to update the configuration to the new style joint axes type of configuration.
- `MACHINE = My Controller` - (string) This is the name of the controller, which is printed out at the top of most graphical interfaces. You can put whatever you want here as long as you make it a single line long.
- `DEBUG = 0` - (u64) Debug level 0 means no messages will be printed when LinuxCNC is run from a terminal. Debug flags are usually only useful to developers. See src/emc/nml_intf/debugflags.h for other settings.
- `NML_FILE = /usr/share/linuxcnc/linuxcnc.nml` - (string) Set this if you want to use a non-default NML configuration file.

[DISPLAY] Abschnitt

Verschiedene Benutzeroberflächen-Programme verwenden unterschiedliche Optionen, und nicht jede Option wird von jeder Benutzeroberfläche unterstützt. Es gibt verschiedene Schnittstellen, wie AXIS, GMOCCAPY, Touchy, QtVCP's QtDragon und Gscreen. AXIS ist eine Schnittstelle für die Verwendung mit normalen Computern und Monitoren, Touchy ist für die Verwendung mit Touchscreens. GMOCCAPY kann in beide Arten verwendet werden und bietet auch viele Anschlüsse für Hardware-Steuerungen. Beschreibungen der Schnittstellen finden Sie im Abschnitt Schnittstellen (engl. Interfaces) des Benutzerhandbuchs.

- `DISPLAY = axis` - (string) The file name of the executable providing the user interface to use. Prominent valid options are (all in lower case): `axis`, `touchy`, `gmoccapy`, `gscreen`, `tklinuxcnc`, `qtvcp`, `qtvcp qtdragon` or `qtvcp qtplasmac`.
- `POSITION_OFFSET = RELATIVE` - (enum) The coordinate system (`RELATIVE` or `MACHINE`) to show on the DRO when the user interface starts. The RELATIVE coordinate system reflects the G92 and G5*x* coordinate offsets currently in effect.
- `POSITION_FEEDBACK = COMMANDED` - (enum) The coordinate value (`COMMANDED` or `ACTUAL`) to show on the DRO when the user interface starts. In AXIS this can be changed from the View menu. The COMMANDED position is the position requested by LinuxCNC. The ACTUAL position is the feedback position of the motors if they have feedback like most servo systems. Typically the COMMANDED value is used.
- `DRO_FORMAT_MM = %+08.6f` - (string) Override the default DRO formatting in metric mode (normally 3 decimal places, padded with spaces to 6 digits to the left). The example above will pad with zeros, display 6 decimal digits and force display of a + sign for positive numbers. Formatting follows Python practice: https://docs.python.org/2/library/string.html#format-specification-mini-language . An error will be raised if the format can not accept a floating-point value.
- `DRO_FORMAT_IN = % 4.1f` - (string) Override the default DRO formatting in imperial mode (normally 4 decimal places, padded with spaces to 6 digits to the left). The example above will display only one decimal digit. Formatting follows Python practice: https://docs.python.org/2/library/string.html#format-specification-mini-language . An error will be raised if the format can not accept a floating-point value.

- `CONE_BASESIZE = .25` - (real) Override the default cone/tool base size of .5 in the graphics display. Valid values are between 0.025 and 2.0.
- `DISABLE_CONE_SCALING = TRUE` - (bool) Will override the default behavior of scaling the cone/tool size using the extents of the currently loaded G-code program in the graphics display.
- `GCODE_VIEW_TOOL_MIN_DIA = 2.0` - (real) If the tool diameter is very small, but non-zero then the displayed cylinder may be too small to see. This could happen if the tool diameter was being used as a wear offset. This setting will cause the tool cone to be displayed rather than a tool cylinder.
- `MAX_FEED_OVERRIDE = 1.2` - (real) The maximum feed override the user may select. 1.2 means 120% of the programmed feed rate.
- `MIN_SPINDLE_OVERRIDE = 0.5` - (real) The minimum spindle override the user may select. 0.5 means 50% of the programmed spindle speed. (This is used to set the minimum spindle speed.)
- `MIN_SPINDLE_0_OVERRIDE = 0.5` - (real) The minimum spindle override the user may select. 0.5 means 50% of the programmed spindle speed. (This is used to set the minimum spindle speed.) On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP based user interfaces.
- `MAX_SPINDLE_OVERRIDE = 1.0` - (real)The maximum spindle override the user may select. 1.0 means 100% of the programmed spindle speed.
- `MAX_SPINDLE_0_OVERRIDE = 1.0` - (real)The maximum spindle override the user may select. 1.2 means 120% of the programmed spindle speed. (This is used to set the maximum spindle speed.) On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP based user interfaces.
- `DEFAULT_SPINDLE_SPEED = 100` - Die Standardspindeldrehzahl, wenn die Spindel im manuellen Modus gestartet wird. Wenn diese Einstellung nicht vorhanden ist, wird sie standardmäßig auf 1 U/min für AXIS und 300 U/min für GMOCCAPY gesetzt.
  - *veraltet* - stattdessen den Abschnitt [SPINDLE_n] verwenden
- `DEFAULT_SPINDLE_0_SPEED = 100` - Die Standardspindeldrehzahl, wenn die Spindel im manuellen Modus gestartet wird. Auf der Mehrspindelmaschine gibt es für jede Spindelnummer Einträge. Wird nur von den QtVCP-basierten Benutzeroberflächen verwendet.
  - *deprecated* - stattdessen den Abschnitt [SPINDLE_n] verwenden.
- `SPINDLE_INCREMENT = 200` - Die Schrittweite, die verwendet wird, wenn man auf die Buttons zum Erhöhen/Verringern klickt. Nur genutzt von QtVCP-basierten GUIs.
  - *deprecated* - stattdessen den Abschnitt [SPINDLE_n] verwenden.
- `MIN_SPINDLE_0_SPEED = 1000` - Die Mindestdrehzahl, die manuell ausgewählt werden kann. Bei Mehrspindelmaschinen gibt es Einträge für jede Spindelnummer. Nur genutzt von den QtVCP-basierten GUIs.
  - *deprecated* - stattdessen den Abschnitt [SPINDLE_n] verwenden.
- `MAX_SPINDLE_0_SPEED = 20000` - Die maximale Drehzahl, die manuell ausgewählt werden kann. Bei Mehrspindelmaschinen gibt es Einträge für jede Spindelnummer. Nur genutzt von QtVCP-basierten GUIs.
  - *deprecated* - stattdessen den Abschnitt [SPINDLE_n] verwenden.

- *PROGRAM_PREFIX = ~/linuxcnc/nc_files* - Das Standardverzeichnis für G-Code-Dateien, benannte Unterprogramme und benutzerdefinierte M-Codes. Das Verzeichnis *PROGRAM_PREFIX* wird vor den Verzeichnissen durchsucht, die in *[RS274]SUBROUTINE_PATH* und *[RS274]USER_M_PATH* aufgeführt sind.
- `INTRO_GRAPHIC = emc2.gif` - Das Bild, das auf dem Begrüßungsbildschirm angezeigt wird.
- `INTRO_TIME = 5` - Die maximale Zeit zur Anzeiges des Startbildschirms, in Sekunden.
- `CYCLE_TIME = 100` - Zykluszeit der Anzeige-GUI. Je nach Bildschirm kann dies in Sekunden oder ms (bevorzugt ms) angegeben werden. Dies ist oft die Aktualisierungsrate und nicht die Ruhezeit zwischen den Aktualisierungen. Wenn die Aktualisierungszeit nicht richtig eingestellt ist, kann der Bildschirm nicht mehr reagieren oder sehr ruckartig werden. Ein Wert von 100 ms (0,1 Sekunden) ist eine übliche Einstellung, obwohl auch ein Bereich von 50 bis 200 ms (0,05 bis 0,2 Sekunden) sinnvoll sein kann. Bei einer leistungsschwachen CPU kann eine längere Einstellung eine Verbesserung bewirken. Normalerweise ist die Standardeinstellung in Ordnung.
- `PREVIEW_TIMEOUT = 5` - Timeout (in Sekunden) für das Laden der grafischen Vorschau des G-Codes. Derzeit nur AXIS.
- `TOOL_CHANGE_SECONDS = 20` - (real) How long one M6 takes on this machine, for the program time estimate. The estimate cannot know what a tool change costs - on a manual changer it is however long the operator takes - so it counts zero unless this says otherwise.
- `HOMING_PROMPT = TRUE` - (bool) Will enable showing a prompt message with homing request, when the Power On button is pressed in AXIS GUI. Pressing the "Ok" button in prompt message is equivalent to pressing the "Home All" button(or the Ctrl-HOME key).
- `FOAM_W = 1.5` setzt die Schaum (engl. foam) W Höhe.
- `FOAM_Z = 0` setzt die Schaum Z Höhe.
- `GRAPHICAL_MAX_FILE_SIZE = 20` größte Größe (in Megabytes), die grafisch angezeigt wird. Wenn das Programm größer als diese Einstellung ist, wird eine Begrenzungsbox angezeigt. Voreingestellt sind 20 MB oder 1/4 des Systemspeichers, was kleiner ist. Ein negativer Wert wird als unlimitiert interpretiert.

NOTE Die folgenden [DISPLAY]-Elemente werden von GladeVCP und PyVCP verwendet, siehe den embedding a tab Abschnitt des GladeVCP Kapitels oder das PyVCP Kapitel für weitere Informationen.

- `EMBED_TAB_NAME = GladeVCP Demo`
- `EMBED_TAB_COMMAND = halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x \{XID\} -u ./gladevcp/hitcounter.py ./gladevcp/manual-example.ui`

NOTE Verschiedene Benutzerschnittstellenprogramme verwenden unterschiedliche Optionen, und nicht jede Option wird von jeder Benutzerschnittstelle unterstützt. Siehe AXIS GUI Dokument für AXIS Details. Siehe GMOCCAPY Dokument für Einzelheiten zu GMOCCAPY.

- `DEFAULT_LINEAR_VELOCITY = .25` - Die Standardgeschwindigkeit für lineares Joggen, in Maschineneinheiten pro Sekunde.

- `MIN_VELOCITY = .01` - Der ungefähre niedrigste Wert des Jog-Sliders.
- `MAX_LINEAR_VELOCITY = 1.0` - Die maximale Geschwindigkeit für lineare Jogs, in Maschineneinheiten pro Sekunde.
- `MIN_LINEAR_VELOCITY = .01` - Der annähernd niedrigste Wert des Jog-Sliders.
- `DEFAULT_ANGULAR_VELOCITY = .25` - Die Standard-Geschwindigkeit für Winkelbewegungen, in Maschineneinheiten pro Sekunde.
- `MIN_ANGULAR_VELOCITY = .01` - Der ungefähre niedrigste Wert des Winkelschiebereglers.
- `MAX_ANGULAR_VELOCITY = 1.0` - Die maximale Geschwindigkeit für Winkelbewegungen, in Maschineneinheiten pro Sekunde.
- `INCREMENTS = 1 mm, .5 in, ...` - Definiert die verfügbaren Inkremente für inkrementelles Joggen. Die INCREMENTS können verwendet werden, um die Standardeinstellung zu überschreiben. Die Werte können Dezimalzahlen (z. B. 0.1000 - mit Dezimalpunkt) oder Bruchzahlen (z. B. 1/16) sein, optional gefolgt von einer Einheit (cm, mm, um, inch, in oder mil). Ohne Angabe einer Einheit wird die Maschineneinheit angenommen. Metrische und imperiale Abstände können gemischt werden: INCREMENTS = 1 inch, 1 mil, 1 cm, 1 mm, 1 µm ist eine gültige Eingabe.
- `GRIDS = 10 mm, 1 in, ...` - Definiert die voreingestellten Werte für Gitterlinien. Der Wert wird auf die gleiche Weise interpretiert wie `INCREMENTS`.
- `OPEN_FILE = /full/path/to/file.ngc` - Die Datei, die beim Start von AXIS in der Vorschau angezeigt wird. Verwenden Sie eine leere Zeichenkette "", wird beim Start keine Datei geladen. GMOCCAPY verwendet diese Einstellung nicht, da es einen entsprechenden Eintrag auf seiner Einstellungsseite anbietet.
- `EDITOR = gedit` - Der Editor, der verwendet werden soll, wenn Sie Datei > Bearbeiten wählen, um den G-Code im Menü AXIS zu bearbeiten. Dieser muss konfiguriert werden, damit dieser Menüpunkt funktioniert. Ein anderer gültiger Eintrag ist `gnome-terminal -e vim`. Dieser Eintrag gilt nicht für GMOCCAPY, da GMOCCAPY einen integrierten Editor hat.
- `TOOL_EDITOR = tooledit` - Der Editor, der bei der Bearbeitung der Werkzeugtabelle verwendet wird (zum Beispiel durch Auswahl von "Datei > Werkzeugtabelle bearbeiten..." in AXIS). Andere gültige Einträge sind `gedit`, `gnome-terminal -e vim`, und `gvim`. Dieser Eintrag gilt nicht für GMOCCAPY, da GMOCCAPY einen integrierten Editor hat.
- `PYVCP = /filename.xml` - Die PyVCP-Panel-Beschreibungsdatei. Siehe das PyVCP-Kapitel für weitere Informationen.
- `PYVCP_POSITION = BOTTOM` - Die Position des PyVCP-Panels in der AXIS-Benutzeroberfläche. Wird diese Variable weggelassen, dann wird das Panel standardmäßig auf der rechten Seite platziert. Die einzige gültige Alternative ist `BOTTOM` (engl. für unten). Weitere Informationen finden Sie im PyVCP-Kapitel.
- `LATHE = 1` - (bool) Causes axis to use "lathe mode" with a top view and with Radius and Diameter on the DRO.
- `BACK_TOOL_LATHE = 1` - (bool) Causes axis to use "back tool lathe mode" with inverted X axis.
- `FOAM = 1` - (bool) Causes axis to change the display for foam-cutter mode.
- `GEOMETRIE = XYZABCUVW` - Steuert die **Vorschau** und **Hintergrunddarstellung** der Bewegung. Dieses Element besteht aus einer Folge von Achsenbuchstaben und Steuerzeichen, denen optional

ein "-" Zeichen vorangestellt ist:

1. Die Buchstaben X, Y, Z geben die Verschiebung entlang der genannten Koordinate an.
2. Die Buchstaben A, B, C bezeichnen die Drehung um die entsprechenden Achsen X, Y, Z.
3. Die Buchstaben U, V, W geben die Verschiebung entlang der entsprechenden Achsen X, Y, Z an.
4. Jeder angegebene Buchstabe muss in `[TRAJ]COORDINATES` vorkommen, um eine Wirkung zu haben.
5. Ein "**-**" Zeichen vor einem beliebigen Buchstaben kehrt die Richtung der Operation um.
6. Die Translations- und Rotationsoperationen werden **von rechts nach links** ausgewertet. Die Verwendung von `GEOMETRY=XYZBC` gibt also eine C-Drehung gefolgt von einer B-Drehung gefolgt von einem Versatz von Z, Y, X an. Die Reihenfolge der aufeinanderfolgenden Versatz-Buchstaben hat keine Auswirkung.
7. Die richtige GEOMETRY-Zeichenkette hängt von der Maschinenkonfiguration und der zur Steuerung verwendeten Kinematik ab. Die Reihenfolge der Buchstaben ist wichtig. Zum Beispiel ist eine Drehung um C und dann B etwas anderes als eine Drehung um B gefolgt von einer um C.
8. Drehungen werden standardmäßig in Bezug auf den Maschinenursprung angewendet. Beispiel: `GEOMETRIE=CXYZ` verschiebt den Kontrollpunkt zunächst nach X, Y, Z und führt dann eine C-Drehung um die Z-Achse aus, die auf den Maschinenursprung zentriert ist.
9. Beispiel für UVW-Verschiebung: `GEOMETRY=XYZUVW` bewirkt, dass UVW im Koordinatensystem des Werkzeugs und XYZ im Koordinatensystem des Materials verschoben wird.
10. Schaumstoff-(engl. foam-)schneidemaschinen (`FOAM = 1`) sollten "XY;UV" angeben oder den Wert leer lassen, auch wenn dieser Wert derzeit im Schaumstoffschneidemodus ignoriert wird. In einer zukünftigen Version kann definiert werden, was ";" bedeutet, aber wenn dies der Fall ist, wird "XY;UV" dasselbe bedeuten wie die aktuelle Voreinstellung.
11. Experimentell: Wenn das Ausrufezeichen (`!`) in der Zeichenkette GEOMETRY enthalten ist, werden die Anzeigepunkte für A-, B- und C-Drehungen unter Berücksichtigung der durch die Codes G5x und G92 festgelegten X-, Y- und Z-Offsets angezeigt. Beispiel: Verwendung von `GEOMETRY = !CXZ` für eine Maschine mit `[TRAJ]COORDINATES=XZC`. Diese Bestimmung gilt nur für Liveplots - G-Code-Vorschauen sollten mit Null G5x, G92 Offsets durchgeführt werden. Dies kann dadurch erleichtert werden, dass die Offsets in den Programmen nur dann gesetzt werden, wenn die Aufgabe läuft (#<_task> == 1). Wenn beim Start aufgrund von Persistenz Offsets ungleich Null vorhanden sind, sollten die Offsets auf Null gesetzt und die Vorschau neu geladen werden.

NOTE Ist keine `[DISPLAY]GEOMETRY` in der INI-Datei beschrieben, wird ein Standardwert durch das `[DISPLAY]DISPLAY`-GUI-Programm bereitgestellt (normalerweise "XYZABCUVW").

- `ARCDIVISION = 64` - Legt die Qualität der Vorschau von Bögen fest. Bögen werden in der Vorschau in eine Anzahl von geraden Linien unterteilt; ein Halbkreis wird in **ARCDIVISION**-viele Teile unterteilt. Größere Werte führen zu einer genaueren Vorschau, aber auch zu längeren Ladezeiten und einer trägeren Darstellung. Kleinere Werte ergeben eine weniger genaue Vorschau, benötigen aber weniger Zeit zum Laden und können zu einer schnelleren Darstellung führen. Der Standardwert von 64 bedeutet, dass ein Kreis von bis zu 3 Zoll mit einer Genauigkeit von 1 mil (.03%)

angezeigt wird.

- `MDI_HISTORY_FILE =` - Der Name einer lokalen MDI-Verlauf-Datei. Wenn dies nicht angegeben wird, speichert AXIS den MDI-Verlauf in **.axis_mdi_history** im Home-Verzeichnis des Benutzers. Dies ist nützlich, wenn Sie mehrere Konfigurationen auf einem Computer haben.
- `JOG_AXES =` - Die Reihenfolge, in der die JOG-Tasten den Achsenbuchstaben zugewiesen werden. Der linke und der rechte Pfeil werden dem ersten Achsenbuchstaben zugewiesen, Auf und Ab dem zweiten, Seite auf/Seite ab dem dritten und linke und rechte Klammer dem vierten. Wenn keine Angaben gemacht werden, wird die Vorgabe von den Angaben zu `[TRAJ]COORDINATES`, `[DISPLAY]LATHE` und `[DISPLAY]FOAM` bestimmt.
- `JOG_INVERT =` - Für jeden Achsenbuchstaben wird die Schrittrichtung invertiert. Die Voreinstellung ist "X" für Drehmaschinen und sonst leer.

NOTE Die Einstellungen für `JOG_AXES` und `JOG_INVERT` gelten für das Joggen im Weltmodus nach Achsenkoordinatenbuchstaben und sind nach erfolgreicher Referenzfahrt im Weltmodus wirksam. Beim Betrieb im Gelenkmodus vor der Referenzfahrt sind die Tastatur-Jog-Tasten in einer festen Reihenfolge zugewiesen: links/rechts: Gelenk0, auf/ab: Gelenk1, Seite auf/Seite ab: Gelenk2, linke/rechte Klammer: Gelenk3

- `USER_COMMAND_FILE = mycommands.py` - Der Name einer optionalen, konfigurationsspezifischen Python-Datei, die von der AXIS GUI anstelle der benutzerspezifischen Datei `~/.axisrc` bezogen wird.

NOTE Der folgende Abschnitt `[DISPLAY]` (engl. für Anzeige) wird nur von der TKLinuxCNC-Schnittstelle verwendet.

- `HELP_FILE = tklinucnc.txt` - Pfad zur Hilfedatei.

[FILTER] Abschnitt

AXIS und GMOCCAPY haben die Möglichkeit, geladene Dateien durch ein Filterprogramm zu schicken. Dieser Filter kann jede gewünschte Aufgabe erfüllen: Etwas so Einfaches wie sicherzustellen, dass die Datei mit M2 endet, oder etwas so Kompliziertes wie die Erkennung, ob es sich bei der Eingabe um ein Tiefenbild handelt, und die Erzeugung von G-Code zum Fräsen der definierten Form. Der Abschnitt `[FILTER]` der INI-Datei steuert, wie die Filter funktionieren. Schreiben Sie zunächst für jeden Dateityp eine `PROGRAM_EXTENSION`-Zeile. Dann geben Sie das Programm an, das für jeden Dateityp ausgeführt werden soll. Dieses Programm erhält den Namen der Eingabedatei als erstes Argument und muss RS274NGC-Code in die Standardausgabe schreiben. Diese Ausgabe ist das, was im Textbereich angezeigt wird, in der Vorschau im Anzeigebereich, und dann auch von LinuxCNC ausgeführt wird.

- `PROGRAM_EXTENSION = .extension Beschreibung`

Wenn Ihr Postprozessor Dateien in Großbuchstaben ausgibt, sollten Sie die folgende Zeile hinzufügen:

```
PROGRAM_EXTENSION = .NGC XYZ Post Processor
```

Die folgenden Zeilen fügen Unterstützung hinzu für die Bild-zu-G-Code-Konverterung mit LinuxCNC.

```
PROGRAM_EXTENSION = .png,.gif,.jpg # Greyscale Depth Image
  png = image-to-gcode
  gif = image-to-gcode
  jpg = image-to-gcode
```

Ein Beispiel für einen benutzerdefinierten G-Code-Konverter, der sich im Verzeichnis linuxcnc befindet.

```
PROGRAM_EXTENSION = .gcode 3D Printer
  gcode = /home/mill/linuxcnc/convert.py
```

NOTE Die Programmdatei, die mit einer Erweiterung verknüpft ist, muss entweder den vollständigen Pfad zum Programm enthalten oder sich in einem Verzeichnis befinden, das sich im Systempfad befindet.

Es ist auch möglich, einen Interpreter anzugeben:

```
PROGRAM_EXTENSION = .py Python Script
  py = python
```

Auf diese Weise kann jedes Python-Skript geöffnet werden, und seine Ausgabe wird als G-Code behandelt. Ein solches Beispielskript ist unter nc_files/holecircle.py verfügbar. Dieses Skript erzeugt G-Code für das Bohren einer Reihe von Löchern entlang des Umfangs eines Kreises. Viele weitere G-Code Generatoren sind auf der LinuxCNC Wiki Seite https://wiki.linuxcnc.org/.

Python-Filter sollten die Funktion print verwenden, um das Ergebnis an AXIS auszugeben.

Dieses Beispielprogramm filtert eine Datei und fügt eine W-Achse hinzu, die der Z-Achse entspricht. Damit es funktioniert, muss zwischen jedem Achsenwort ein Leerzeichen stehen.

```
#!/usr/bin/env python3

import sys

def main(argv):

  openfile = open(argv[0], 'r')
  file_in = openfile.readlines()
  openfile.close()

  file_out = []
  for line in file_in:
    # print(line)
    if line.find('Z') != -1:
      words = line.rstrip('\n')
      words = words.split(' ')
      newword = ''
      for i in words:
        if i[0] == 'Z':
          newword = 'W'+ i[1:]
```

```
      if len(newword) > 0:
        words.append(newword)
        newline = ' '.join(words)
        file_out.append(newline)
    else:
      file_out.append(line)
  for item in file_out:
    print("%s" % item)

if __name__ == "__main__":
  main(sys.argv[1:])
```

- FILTER_PROGRESS=%d +
  If the environment variable AXIS_PROGRESS_BAR is set, then lines written to stderr of the form above sets the AXIS progress bar to the given percentage. This feature should be used by any filter that runs for a long time.

[RS274NGC] Abschnitt

- PARAMETER_FILE = myfile.var - Die Datei (engl. file), die sich im gleichen Verzeichnis wie die INI-Datei befindet und die vom Interpreter verwendeten Parameter enthält (zwischen den Läufen gespeichert).
- ORIENT_OFFSET = 0 - Eine Gleitkommazahl, die zum R-Wort-Parameter einer M19 Orient Spindle Operation hinzugefügt wird. Wird verwendet, um eine beliebige Nullposition zu definieren, unabhängig von der Ausrichtung der Encoder.
- RS274NGC_STARTUP_CODE = G17 G20 G40 G49 G64 P0.001 G80 G90 G92.1 G94 G97 G98 - Eine Folge von NC-Codes, mit denen der Interpreter initialisiert wird. Dies ist kein Ersatz für die Angabe von modalen G-Codes am Anfang jeder NGC-Datei, da die modalen Codes der Maschinen unterschiedlich sind und durch einen früher in der Sitzung interpretierten G-Code geändert werden können.
- SUBROUTINE_PATH = ncsubroutines:/tmp/testsubs:lathesubs:millsubs - Gibt eine durch Doppelpunkt (:) getrennte Liste von bis zu 10 Verzeichnissen an, die durchsucht werden sollen, wenn Unterprogramme in einer einzigen Datei im G-Code angegeben werden. Diese Verzeichnisse werden nach der Suche in [DISPLAY]PROGRAM_PREFIX (falls angegeben) und vor der Suche in [WIZARD]WIZARD_ROOT (falls angegeben) durchsucht. Die Pfade werden in der Reihenfolge durchsucht, in der sie aufgelistet sind. Die als erste bei der Suche gefundene passende Unterprogrammdatei wird verwendet. Die Verzeichnisse werden relativ zum aktuellen Verzeichnis für die INI-Datei oder als absolute Pfade angegeben. Die Liste darf keine Leerzeichen dazwischen enthalten.
- G64_DEFAULT_TOLERANCE = *n* (Voreinstellung: 0) Voreingestellter Wert für P bei G64 wenn P nicht explizit angegeben ist.
- G64_DEFAULT_NAIVETOLERANCE = *n* (Voreinstellung: 0) Voreingestellter Wert für Q bei G64 wenn Q nicht explizit angegeben ist.
- CENTER_ARC_RADIUS_TOLERANCE_INCH = *n* (Voreinstellung: 0.00005)
- CENTER_ARC_RADIUS_TOLERANCE_MM = *n* (Voreinstellung: 0.00127)
- USER_M_PATH = myfuncs:/tmp/mcodes:experimentalmcodes - Gibt eine Liste von durch

Doppelpunkt (:) getrennten Verzeichnissen für benutzerdefinierte Funktionen an. Die Verzeichnisse werden relativ zum aktuellen Verzeichnis für die INI-Datei oder als absolute Pfade angegeben. Die Liste darf keine Leerzeichen dazwischen enthalten.

Es wird nach jeder möglichen benutzerdefinierten Funktion gesucht, typischerweise (M100-M199). Die Reihenfolge der Suche ist:

1. `[DISPLAY]PROGRAM_PREFIX` (falls angegeben)
2. Wenn `[DISPLAY]PROGRAM_PREFIX` nicht angegeben ist, wird der Standardspeicherort gesucht: nc_files
3. Dann wird jedes Verzeichnis in der Liste `[RS274NGC]USER_M_PATH` durchsucht.

   Für jeden M1*xx* wird der erste bei der Suche gefundene ausführbare M1*xx* verwendet.

NOTE	Die maximale Anzahl der `USER_M_PATH`-Verzeichnisse wird zur Kompilierzeit festgelegt (Typ: `USER_DEFINED_FUNCTION_MAX_DIRS == 5`).

- `INI_VARS = 1` (Standardwert: 1)
  Erlaubt G-Code-Programmen, Werte aus der INI-Datei im Format #<_ini[section]name> zu lesen. Siehe G-Code Parameter.
- `HAL_PIN_VARS = 1` (Voreinstellung: 1)
  Erlaubt G-Code-Programmen das Lesen der Werte von HAL-Pins unter Verwendung des Formats #<_hal[HAL item]>. Der Zugriff auf die Variablen ist schreibgeschützt. Siehe G-Code Parameter für weitere Details und eine wichtige Warnung.
- `RETAIN_G43 = 0` (Voreinstellung: 0)
  Wenn diese Option eingestellt ist, können Sie G43 nach dem Laden des ersten Werkzeugs einschalten und müssen sich dann nicht mehr um das Programm kümmern. Wenn Sie schließlich das letzte Werkzeug entladen, wird der G43-Modus deaktiviert.
- `OWORD_NARGS = 0` (Voreinstellung: 0)
  Wenn diese Funktion aktiviert ist, kann ein aufgerufenes Unterprogramm die Anzahl der tatsächlich übergebenen Positionsparameter ermitteln, indem es den `#<n_args>` Parameter untersucht.
- `NO_DOWNCASE_OWORD = 0` (Voreinstellung: 0)
  Groß- und Kleinschreibung in O-Wort-Namen innerhalb von Kommentaren beibehalten, falls gesetzt, ermöglicht das Lesen von HAL-Elementen mit gemischter Groß- und Kleinschreibung in strukturierten Kommentaren wie `(debug, #<_hal[MixedCaseItem])`.
- `OWORD_WARNONLY = 0` (Voreinstellung: 0)
  Warnung statt Fehler bei Fehlern in O-Wort-Unterprogrammen.
- `DISABLE_G92_PERSISTENCE = 0` (bool, Default: 0) Allow to clear the G92 offset automatically when config start-up.
- `DISABLE_AUTO_G54 = 0` (bool, Default: 0)
  When set M2 and M30 will no longer automatically reset the active WCS to *G54*.
- `RETAIN_WORK_PLANE = 0` (bool, Default: 0)
  When set M2, M30 and an abort no longer cancel a tilted work plane. A restart still does.

- `DISABLE_FANUC_STYLE_SUB = 0` (Standardwert: 0) Wenn es einen Grund gibt, Fanuc-Unterprogramme zu deaktivieren, setzen Sie diesen Wert auf 1.
- *G73_PECK_CLEARANCE = .020* (Voreinstellung: Metrische Maschine: 1 mm, imperiale Maschine: .05 Zoll) Splitter/Span (chip)-Back-off-Abstand in Maschineneinheiten
- *G83_PECK_CLEARANCE = .020* (Voreinstellung: Metrische Maschine: 1mm, imperiale Maschine: .050 Zoll) Freiraumabstand (engl. clearance distance) von der letzten Eintauchtiefe (engl. feed depth), wenn die Maschine schnell zum Boden des Lochs zurückfährt, in Maschineneinheiten.

NOTE

Die oben genannten sechs Optionen wurden durch die `FEATURES` Bitmaske in Versionen von LinuxCNC vor 2.8 gesteuert. Dieser INI-Tag wird nicht mehr funktionieren.
Als Referenz:

```
FEATURES & 0x1  -> RETAIN_G43
FEATURES & 0x2  -> OWORD_NARGS
FEATURES & 0x4  -> INI_VARS
FEATURES & 0x8  -> HAL_PIN_VARS
FEATURES & 0x10 -> NO_DOWNCASE_OWORD
FEATURES & 0x20 -> OWORD_WARNONLY
```

NOTE

`[WIZARD]WIZARD_ROOT` ist ein gültiger Suchpfad, aber der Assistent ist noch nicht vollständig implementiert und die Ergebnisse seiner Verwendung sind unvorhersehbar.

- `LOG_LEVEL = 0` Bestimmt den log_level (Voreinstellung: 0)
- *LOG_FILE = file-name.log*
  Zur Angabe der Datei, die für die Protokollierung der Daten verwendet wird.
- `REMAP=M400 modalgroup=10 argspec=Pq ngc=myprocedure` Siehe das Remap Erweiterung von G-Code Kapitel für Details.
- `ON_ABORT_COMMAND=O <on_abort> call` Siehe das Remap Erweiterung von G-Code Kapitel für Details.

[EMCMOT] Abschnitt

Dieser Abschnitt ist ein benutzerdefinierter Abschnitt und wird nicht von LinuxCNC direkt verwendet. Die meisten Konfigurationen verwenden Werte aus diesem Abschnitt, um den Motion-Controller zu laden. Für weitere Informationen über die Motion-Controller siehe den Abschnitt zu Motion.

- `EMCMOT = motmod` - hier wird in der Regel der Name des Motion Controllers verwendet.
- `BASE_PERIOD = 50000` - die *Basis* (engl. base) Taskdauer in Nanosekunden.
- `SERVO_PERIOD = 1000000` - Dies ist die "Servo" Task Periode in Nanosekunden.
- `TRAJ_PERIOD = 100000` - Dies ist die Aufgabenperiode (engl. *task period*) des Trajektorienplaners (engl. *trajectory planner*) in Nanosekunden.
- `COMM_TIMEOUT = 1.0` - Anzahl der Sekunden, die gewartet wird, bis Motion (der Echtzeitteil des Bewegungssteuerungssystems) den Empfang von Nachrichten von Task (dem Nicht-Echtzeitteil des Bewegungssteuerungssystems) bestätigt.

- `HOMEMOD = alternate_homing_module` [home_parms=value] The HOMEMOD variable is optional. If specified, use a specified (user-built) module instead of the default (homemod). The homecomp(9) component provides a buildable template for custom homing modules. Module parameters (home_parms) may be included if supported by the named module. The setting may be overridden from the command line using the -m option ($ linuxcnc -h).

[TASK] Abschnitt

- *TASK* = *milltask* - Gibt den Namen der ausführbaren Datei *task* an. Die ausführbare Datei "task" führt verschiedene Dinge aus, wie beispielsweise
  - Kommunikation mit den Benutzeroberflächen über NML,
  - mit dem Echtzeit-Bewegungsplaner über einen nicht-HAL-geteilten Speicher kommunizieren und
  - G-Code interpretieren. Derzeit gibt es nur eine ausführbare Aufgabe, die für 99,9 % der Benutzer sinnvoll ist: milltask.
- `CYCLE_TIME = 0.010` - The period, in seconds, at which TASK will run. This parameter affects the polling interval when waiting for motion to complete, when executing a pause instruction, and when accepting a command from a user interface. There is usually no need to change this number. Defaults to 0.100 if omitted.

[HAL] Abschnitt

- `HALFILE = beispiel.hal` - Führt die Datei *beispiel.hal* beim Start aus.

  Wenn `HALFILE` mehrfach angegeben wird, werden die Dateien in der Reihenfolge interpretiert, in der sie in der INI-Datei erscheinen. HAL-Dateien sind beschreibend, die Ausführung dessen, was in HAL-Dateien beschrieben ist, wird durch die Threads ausgelöst, in die Funktionen eingebettet sind, nicht durch das Lesen der HAL-Datei. Fast alle Konfigurationen haben mindestens eine `HALFILE`, und Steppersysteme haben typischerweise zwei solcher Dateien, d.h. eine zur Spezifikation der allgemeinen Stepperkonfiguration (*core_stepper.hal*) und eine zur Spezifikation der Pinbelegung der Maschine (*xxx_pinout.hal*).

  HAL-Dateien, die in der Variablen `HALFILES` angegeben sind, werden durch eine Suche gefunden. Wenn die benannte Datei in dem Verzeichnis gefunden wird, das die INI-Datei enthält, wird sie verwendet. Wird die genannte Datei nicht in diesem INI-Verzeichnis gefunden, wird eine Systembibliothek mit HAL-Dateien durchsucht.

  Wenn LinuxCNC mit dem Skript `linuxcnc` unter Verwendung der Option "`-H` *dirname*" gestartet wird, dann wird der angegebene Verzeichnisname der oben beschriebenen Suche vorangestellt, so dass *dirname* zuerst durchsucht wird. Die Option "`-H` *dirname*" kann mehr als einmal angegeben werden, die Verzeichnisse werden in der Reihenfolge vorangestellt.

  Eine HALFILE kann auch als absoluter Pfad angegeben werden (wenn der Name mit einem / Zeichen beginnt). Absolute Pfade werden nicht empfohlen, da ihre Verwendung das Verschieben von Konfigurationen einschränken kann.

- `HALFILE = texample.tcl` [*arg1* [*arg2*] ...] - Führt die tcl Datei *texample.tcl* beim Start mit arg1,

arg2, etc als ::argv Liste aus. Dateien mit dem Suffix .tcl werden wie oben beschrieben verarbeitet, verwenden aber haltcl zur Verarbeitung. Weitere Informationen finden Sie im Kapitel HALTCL.

- `HALFILE = LIB:sys_example.hal` - Führt die Systembibliotheksdatei `sys_example.hal` beim Starten aus. Die explizite Verwendung des Präfixes LIB: bewirkt, dass die Systembibliothek HALFILE verwendet wird, ohne das Verzeichnis der INI-Datei zu durchsuchen.
- `HALFILE = LIB:sys_texample.tcl` [*arg1* [*arg2* ...]]' - Führt die Systembibliotheksdatei `sys_texample.tcl` beim Starten aus. Die explizite Verwendung des Präfixes LIB: bewirkt, dass die Systembibliothek HALFILE verwendet wird, ohne dass das Verzeichnis der INI-Datei durchsucht wird.

HALFILE items specify files that loadrt HAL components and make signal connections between component pins. Common mistakes are:

1. Wegfall der addf-Anweisung, die erforderlich ist, um die Funktion(en) einer Komponente zu einem Thread hinzuzufügen,
2. unvollständige Signal(netz)bezeichner.

Das Weglassen der erforderlichen addf-Anweisungen ist fast immer ein Fehler. Signale umfassen in der Regel eine oder mehrere Eingangsverbindungen und einen einzelnen Ausgang (beides ist jedoch nicht unbedingt erforderlich). Eine Systembibliotheksdatei wird bereitgestellt, um diese Bedingungen zu prüfen und auf stdout und in einer Pop-up-GUI zu melden:

```
HALFILE = LIB:halcheck.tcl [nopopup]
```

NOTE Die Zeile LIB:halcheck.tcl sollte die letzte [HAL]HALFILE sein. Geben Sie die Option *nopopup* an, um die Popup-Meldung zu unterdrücken und einen sofortigen Start zu ermöglichen. Die über eine POSTGUI_HALFILE hergestellen Verbindungen werden nicht geprüft.

- TWOPASS = ON" - Verwenden Sie die Verarbeitung in zwei Durchgängen zum Laden von HAL-Komponenten. Bei der TWOPASS-Verarbeitung werden die Zeilen der in `[HAL]HALFILE` angegebenen Dateien in zwei Durchgängen verarbeitet. Im ersten Durchgang (pass0) werden alle HALFILES gelesen und mehrere Auftritte von loadrt- und loadusr-Befehlen kumuliert. Diese kumulierten Ladebefehle werden am Ende von pass0 ausgeführt. Durch diese Akkumulation können Ladezeilen für ein bestimmtes Bauteil mehr als einmal angegeben werden (vorausgesetzt, die verwendeten names= Namen sind bei jeder Verwendung eindeutig). Im zweiten Durchlauf (pass1) werden die HALFILES erneut eingelesen und alle Befehle außer den zuvor ausgeführten Ladebefehlen ausgeführt.
- `TWOPASS = nodelete verbose` - Das `TWOPASS` kann mit jeder Zeichenkette, die nicht Null ist, aktiviert werden, einschließlich der Schlüsselwörter verbose und nodelete. Das Schlüsselwort verbose bewirkt die Ausgabe von Details auf stdout. Das Schlüsselwort nodelete bewahrt temporäre Dateien in /tmp.

Weitere Informationen finden Sie im Kapitel HAL TWOPASS.

- `HALCMD = command` - Führt *command* als einzelnen HAL-Befehl aus. Wenn `HALCMD` mehrfach

angegeben wird, werden die Befehle in der Reihenfolge ausgeführt, in der sie in der INI-Datei stehen. Die HALCMD-Zeilen werden nach allen HALFILE-Zeilen ausgeführt.

- SHUTDOWN = shutdown.hal - Führt die Datei *shutdown.hal* aus, wenn LinuxCNC beendet wird. Abhängig von den verwendeten Hardware-Treibern, kann dies es möglich machen, Ausgänge auf definierte Werte zu setzen, wenn LinuxCNC normal beendet wird. Da es jedoch keine Garantie dafür gibt, dass diese Datei ausgeführt wird (z.B. im Falle eines Computerabsturzes), ist sie kein Ersatz für eine korrekte physische E-Stop-Kette oder andere Schutzmaßnahmen gegen Softwarefehler.
- POSTGUI_HALFILE = example2.hal - Führen Sie *example2.hal* aus, nachdem die GUI ihre HAL-Pins erstellt hat. Einige GUIs erzeugen HAL-Pins und unterstützen die Verwendung einer Postgui-HAL-Datei, um sie zu nutzen. Zu den GUIs, die postgui halffiles unterstützen, gehören Touchy, AXIS, Gscreen und GMOCCAPY.
  Siehe Abschnitt PyVCP with AXIS für weitere Informationen.
- HALUI = halui - fügt die HAL-Benutzerschnittstellen-Pins hinzu.
  Für weitere Informationen siehe das Kapitel HAL Benutzerschnittstelle (engl. HAL user interface).

[HALUI] Abschnitt

- MDI_COMMAND = G53 G0 X0 Y0 Z0 - Ein MDI-Befehl kann mit halui.mdi-command-00 ausgeführt werden. Erhöhen Sie die Zahl für jeden im Abschnitt [HALUI] aufgeführten Befehl. Es ist auch möglich, Unterroutinen (engl. subroutines) zu starten. MDI_COMMAND = o<deineunterroutine> CALL [#<deinevariable>]

[APPLICATIONS] Abschnitt

LinuxCNC kann andere Anwendungen starten, bevor die angegebene Benutzeroberfläche gestartet wird. Die Anwendungen können nach einer bestimmten Verzögerung gestartet werden, um GUI-abhängige Aktionen zu ermöglichen (wie das Erstellen von GUI-spezifischen HAL-Pins).

- DELAY = *wert* - Dauer in Sekunden, die vor dem Start anderer Anwendungen gewartet wird. Eine Verzögerung kann erforderlich sein, wenn eine Anwendung Abhängigkeiten von [HAL]POSTGUI_HALFILE-Aktionen oder von durch das GUI erstellten HAL Pins hat (Voreinstellung: DELAY=0).
- APP = *appname* [arg1 [arg2 ...]]' - Zu startende Anwendung. Diese Angabe kann mehrfach enthalten sein. Der Anwendungsname kann explizit als absoluter oder mit Tilde angegebener Dateiname (erstes Zeichen ist / oder ~), als relativer Dateiname (erste Zeichen des Dateinamens sind ./) oder als Datei im INI-Verzeichnis angegeben werden. Wird keine ausführbare Datei mit diesen Namen gefunden, wird die Anwendung über die Benutzersuche PATH gefunden.
  Beispiele:
  - Simulation von Eingängen an HAL-Pins zum Testen (unter Verwendung von sim_pin — einer einfachen Benutzeroberfläche zum Setzen von Eingängen an Parameter, nicht angeschlossene Pins oder Signale ohne Schreiber):

```
APP = sim_pin motion.probe-input halui.abort motion.analog-in-00
```

- Rufen Sie halshow mit einer zuvor gespeicherten Beobachtungsliste auf. Da LinuxCNC das

Arbeitsverzeichnis auf das Verzeichnis für die INI-Datei setzt, können Sie auf Dateien in diesem Verzeichnis verweisen (Beispiel: my.halshow):

```
APP = halshow my.halshow
```

- Alternativ kann auch eine Watchlist-Datei mit einem vollständigen Pfadnamen angegeben werden:

```
APP = halshow ~/saved_shows/spindle.halshow
```

- Öffnen Sie halscope mit einer zuvor gespeicherten Konfiguration:

```
APP = halscope -i my.halscope
```

[TRAJ] Abschnitt

WARNING	Die neue Trajectorien Planer (TP) (engl. trajectory planner) ist standardmäßig aktiv. Wenn Sie keine TP-Einstellungen in Ihrem [TRAJ]-Abschnitt haben - LinuxCNC standardmäßig auf: ARC_BLEND_ENABLE = 1 ARC_BLEND_FALLBACK_ENABLE = 0 ARC_BLEND_OPTIMIZATION_DEPTH = 50 ARC_BLEND_GAP_CYCLES = 4 ARC_BLEND_RAMP_FREQ = 100

Der Abschnitt [TRAJ] enthält allgemeine Parameter für das Trajektorienplanungsmodul in *motion*.

- `ARC_BLEND_ENABLE = 1` - (bool) Turn on new TP. If set to 0 TP uses parabolic blending (1 segment look ahead) (Default: 1).
- `ARC_BLEND_FALLBACK_ENABLE = 0` - (bool) Optionally fall back to parabolic blends if the estimated speed is faster. However, this estimate is rough, and it seems that just disabling it gives better performance (Default: 0).
- `ARC_BLEND_OPTIMIZATION_DEPTH = 50` - Vorausschauende Tiefe in Anzahl der Segmente.

  Um dies ein wenig zu erweitern, können Sie diesen Wert einigermaßen willkürlich wählen. Hier's eine Formel, um zu schätzen, wie viel *Tiefe* Sie für eine bestimmte Konfiguration benötigen:

  ```
  # n = v_max / (2.0 * a_max * t_c)
  # wobei:
  # n = Optimierungstiefe
  # v_max = maximale Achsengeschwindigkeit (UU / sec)
  # a_max = maximale Achsenbeschleunigung (UU / sec)
  # t_c = Servo-Periode (Sekunden)
  ```

  So würde eine Maschine mit einer maximalen Achsengeschwindigkeit von 10 IPS, einer maximalen Beschleunigung von 100 IPS^2 und einer Servoperiode von 0,001 s benötigen:

  10 / (2,0 * 100 * 0,001) = 50 Segmente, um immer die maximale Geschwindigkeit entlang der

schnellsten Achse zu erreichen.

In der Praxis ist die Einstellung dieser Zahl nicht so wichtig, da die Vorausschau selten die volle Tiefe benötigt, es sei denn, Sie haben viele sehr kurze Segmente. Wenn Sie beim Testen merkwürdige Verlangsamungen bemerken und nicht herausfinden können, woher sie kommen, versuchen Sie zunächst, diese Tiefe mit Hilfe der obigen Formel zu erhöhen.

Wenn Sie immer noch seltsame Verlangsamungen feststellen, kann das daran liegen, dass Sie kurze Segmente im Programm haben. Wenn dies der Fall ist, versuchen Sie, eine kleine Toleranz für die naive CAM-Erkennung hinzuzufügen. Eine gute Faustregel ist diese:

```
# min_length ~= v_req * t_c
# wobei:
# v_req = gewünschte Geschwindigkeit in UU / sec
# t_c = Servoperiode (Sekunden)
```

Wenn Sie eine Bahn mit 1 IPS = 60 IPM fahren wollen und Ihre Servoperiode 0,001 s beträgt, dann verlangsamen alle Segmente, die kürzer als min_length sind, die Bahn. Wenn Sie die Naive CAM-Toleranz auf etwa diese Mindestlänge einstellen, werden zu kurze Segmente zusammengefasst, um diesen Engpass zu beseitigen. Wenn Sie die Toleranz zu hoch einstellen, bedeutet das natürlich große Pfadabweichungen, so dass Sie ein wenig damit spielen müssen, um einen guten Wert zu finden. Ich würde mit 1/2 der Mindestlänge beginnen und dann nach Bedarf erhöhen.

- `ARC_BLEND_GAP_CYCLES = 4` Wie kurz das vorherige Segment sein muss, bevor es vom Trajektorienplaner *verbraucht* wird.

  Bei einer Kreisbogenüberblendung bleiben oft kurze Liniensegmente zwischen den Überblendungen übrig. Da die Geometrie kreisförmig sein muss, können wir nicht eine ganze Linie überblenden, wenn die nächste etwas kürzer ist. Da der Trajektorienplaner jedes Segment mindestens einmal berühren muss, bedeutet dies, dass sehr kleine Segmente die Dinge erheblich verlangsamen. Meine Lösung für dieses Problem besteht darin, das kurze Segment zu "verbrauchen", indem ich es zu einem Teil des Überblendungsbogens mache. Da die Linie und die Überblendung ein einziges Segment sind, müssen wir nicht langsamer werden, um das sehr kurze Segment zu treffen. Wahrscheinlich brauchen Sie diese Einstellung nicht zu ändern.

- `ARC_BLEND_RAMP_FREQ = 20` - Dies ist eine *cutoff* Frequenz für die Verwendung von rampenförmigen Geschwindigkeiten.

  Ramped velocity bedeutet in diesem Fall eine konstante Beschleunigung über das gesamte Segment. Dies ist weniger optimal als ein trapezförmiges Geschwindigkeitsprofil, da die Beschleunigung nicht maximiert wird. Wenn das Segment jedoch kurz genug ist, bleibt nicht genug Zeit, um viel zu beschleunigen, bevor wir das nächste Segment erreichen. Erinnern Sie sich an die kurzen Streckenabschnitte aus dem vorherigen Beispiel. Da es sich um Linien handelt, gibt es keine Kurvenbeschleunigung, wir können also bis zur gewünschten Geschwindigkeit beschleunigen. Wenn sich diese Linie jedoch zwischen zwei Bögen befindet, muss sie schnell wieder abbremsen, um innerhalb der Höchstgeschwindigkeit des nächsten Segments zu liegen. Das bedeutet, dass wir eine Beschleunigungsspitze und dann eine Abbremsspitze haben, was zu einem großen Ruck führt und nur einen geringen Leistungsgewinn bringt. Mit dieser Einstellung lässt sich dieses Ruckeln bei kurzen Segmenten vermeiden.

Grundsätzlich gilt: Wird ein Segment in weniger als 1 / ARC_BLEND_RAMP_FREQ abgeschlossen, dann wird kein trapezförmiges Geschwindigkeitsprofil für dieses Segment verwendet, sondern eine konstante Beschleunigung. (Die Einstellung `ARC_BLEND_RAMP_FREQ = 1000` ist gleichbedeutend mit der Verwendung einer trapezförmigen Beschleunigung, wenn die Servoschleife 1 kHz hat).

Sie können den schlimmsten Leistungsverlust charakterisieren, indem Sie die maximale Geschwindigkeit eines trapezförmiges Profils vergleichen mit der durch eine Rampe zu erreichenden:

```
# v_ripple = a_max / (4.0 * f)
# wobei:
# v_ripple = durchschnittliche Geschwindigkeit "Verlust" aufgrund von Rampen
# a_max = maximale Achsenbeschleunigung
# f = Grenzfrequenz aus INI
```

Für die oben genannte Maschine beträgt die Restwelligkeit bei einer Grenzfrequenz von 20 Hz 100 / (4 * 20) = 1,25 IPS. Dies erscheint hoch, aber bedenken Sie, dass es sich nur um eine Worst-Case-Schätzung handelt. In Wirklichkeit wird das trapezförmige Bewegungsprofil durch andere Faktoren wie die normale Beschleunigung oder die gewünschte Geschwindigkeit begrenzt, so dass der tatsächliche Leistungsverlust viel geringer sein dürfte. Eine Erhöhung der Grenzfrequenz kann mehr Leistung herausholen, macht aber die Bewegung aufgrund von Beschleunigungssprüngen unruhiger. Ein Wert im Bereich von 20 Hz bis 200 Hz sollte für den Anfang angemessen sein.

Und schließlich können Sie einen Werkzeugweg mit vielen kleinen, engen Kurven nicht beschleunigen, da Sie durch die Kurvenbeschleunigung eingeschränkt sind.

- `SPINDLES = 3` - Die Anzahl der zu unterstützenden Spindeln. Diese Zahl muss unbedingt mit dem Parameter "num_spindles" übereinstimmen, der an das Bewegungsmodul übergeben wird.
- `COORDINATES = X Y Z` - Die Namen der gesteuerten Achsen. Nur X, Y, Z, A, B, C, U, V, W sind gültig. Nur die in *COORDINATES* genannten Achsen werden im G-Code akzeptiert. Es ist erlaubt, einen Achsennamen mehr als einmal zu schreiben (z.B. X Y Y Z für eine Gantry-Maschine). Bei der üblichen *trivkins-Kinematik* werden die Gelenknummern der Reihe nach gemäß dem trivkins-Parameter *coordinates*= vergeben. Für trivkins *coordinates=xz* entspricht joint0 also X und joint1 entspricht Z. Informationen zu trivkins und anderen Kinematikmodulen finden Sie in der Manpage Kinematics (*$ man kins*).
- `FEED_AXES = X Y Z` - The axes the programmed feed rate F is measured along. All of them must be `LINEAR` axes; the default is X Y Z. A move goes at F along the straight line through the feed axes that move, and every other axis in the move arrives at the same time. A move in which no feed axis moves goes at F along the other linear axes that move, and a move of angular axes only goes at F in degrees per minute. With the defaults this is the RS274NGC rule: X Y Z, else U V W, else A B C. With `FEED_AXES = X Y Z U`, for example, a move of X and U together goes at F along the line through both. The maximum velocity slider of the GUIs caps the rate along that same line, and does not cap a move of angular axes only.
- `LINEAR_UNITS =` <*units*> - Gibt die *Maschineneinheiten* für lineare Achsen an. Mögliche Auswahlen sind mm oder inch (engl. für Zoll). Dies hat keinen Einfluss auf die linearen Einheiten im NC-Code (die Wörter G20 und G21 tun dies).
- `ANGULAR_UNITS =` <*units*> - Specifies the *machine units* for rotational axes. Possible choices are *deg*,

degree (360 per circle), *rad*, *radian* (2*π per circle), *grad*, or *gon* (400 per circle). This does not affect the angular units of NC code. In RS274NGC, the words of an ANGULAR axis, A, B and C by default, are always expressed in degrees.

- DEFAULT_LINEAR_VELOCITY = 0.0167 - Die anfängliche Geschwindigkeit für Jogs von Linearachsen, in Maschineneinheiten pro Sekunde. Der in *Axis* angezeigte Wert entspricht den Maschineneinheiten pro Minute.
- DEFAULT_LINEAR_ACCELERATION = 2.0 - In Maschinen mit nicht trivialer Kinematik, die Beschleunigung für "teleop" (kartesischer Raum) Jogging, in *Maschineneinheiten* pro Sekunde pro Sekunde.
- MAX_LINEAR_VELOCITY = 5.0 - Die maximale Geschwindigkeit für eine beliebige Achse oder koordinierte Bewegung, in *Maschineneinheiten* pro Sekunde. Der angezeigte Wert entspricht 300 Einheiten pro Minute.
- MAX_LINEAR_ACCELERATION = 20.0 - Die maximale Beschleunigung für jede Achse oder koordinierte Achsenbewegung, in *Maschineneinheiten* pro Sekunde.
- PLANNER_TYPE = 0 - Selects the trajectory planner type: 0 = trapezoidal (default), 1 = S-curve with jerk limiting. S-curve planning is only active when PLANNER_TYPE = 1 AND MAX_LINEAR_JERK is 1.0 or above; a smaller value falls back to trapezoidal, with an operator message. The planner can be changed at runtime via the ini.traj-planner-type HAL pin or the G64 R word; a switch requested while the machine is moving is deferred and applied when motion next becomes idle (never aborts), with an operator message.
- SMOOTH_PLANNER = 1 - Which planner a G64 R greater than zero ("smooth") request resolves to. Part programs only request trapezoidal (R0) or smooth (R greater than 0); this key names the implementation, so programs stay valid when the machine's smooth planner changes. 0 disables G64 R smooth requests (they are refused with an operator error). Defaults to PLANNER_TYPE when that is non-zero, else 1. Currently the only smooth planner is type 1. A missing or invalid MAX_LINEAR_JERK (below 1) also disables smooth requests. Note that the ini.traj-planner-type HAL pin is the machine-side surface and writes planner numbers directly; it bypasses this resolution and its guards.
- SCURVE_PEAK_SCALE = 1.0 - Scales the S-curve rest-to-rest peak velocity used by the corner look-ahead, i.e. how aggressively the machine may corner within its jerk and acceleration limits. 1.0 (the default) allows the full jerk-feasible corner speed; lower values corner more gently (0.5 matches the initial S-curve release). Clamped to 0.1..1.0. Tunable at runtime via the ini.traj-scurve-peak-scale HAL pin or the G64 R word.
- MAX_LINEAR_JERK = 10000.0 - The maximum jerk (rate of change of acceleration) for coordinated moves, in *machine units* per second cubed. Default is 1e9 (1 billion) if not specified, which effectively disables jerk limiting while avoiding numerical instability. Values are clamped to a maximum of 1e9 to prevent numerical issues in S-curve calculations. When PLANNER_TYPE = 1, this enables S-curve trajectory planning. Note: Not specifying MAX_LINEAR_JERK (defaulting to 1e9) produces motion similar to trapezoidal planning (PLANNER_TYPE = 0) but not identical, as extremely high jerk still uses S-curve calculations.
- POSITION_FILE = *position.txt* - Wenn auf einen nicht leeren Wert gesetzt, werden die Gelenkpositionen zwischen den Läufen in dieser Datei gespeichert. Dadurch kann die Maschine mit denselben Koordinaten starten, die sie beim Herunterfahren hatte. Dabei wird davon ausgegangen, dass die Maschine im ausgeschalteten Zustand nicht bewegt wurde. Wenn nicht gesetzt, sind gemeinsame Positionen nicht gespeichert und wird bei 0 beginnen jedes Mal, wenn LinuxCNC

gestartet wird. Dies kann auf kleineren Maschinen ohne Home-Schalter helfen. Bei Verwendung der Mesa Resolver-Schnittstelle kann diese Datei verwendet werden, um absolute Encoder zu emulieren und die Notwendigkeit für die Referenzfahrt (ohne Verlust der Genauigkeit) zu beseitigen. Siehe die hostmot2 Manpage für weitere Details.

- `NO_FORCE_HOMING = 1` - (bool) The default behavior is for LinuxCNC to force the user to home the machine before any MDI command or a program is run. Normally, only jogging is allowed before homing. For configurations using identity kinematics, setting `NO_FORCE_HOMING = 1` allows the user to make MDI moves and run programs without homing the machine first. Interfaces using identity kinematics without homing ability will need to have this option set to 1.

WARNING	LinuxCNC kennt die Grenzen (engl. limits) der Gelenke nicht, wenn `NO_FORCE_HOMING = 1` gesetzt.

- `HOME = 0 0 0 0 0 0 0 0 0` - Welt-Referenzpunkt-(engl. home)-Position, die für Kinematik-Module benötigt wird, um die Weltkoordinaten mit kinematicsForward() berechnen, wenn sie vom Joint- in den Teleop-Modus wechseln. Es können bis zu neun Koordinatenwerte (X, Y, Z, A, B, C, U, V, W) angegeben werden, unbenutzte Nachkommastellen können weggelassen werden. Dieser Wert wird nur für Maschinen mit nicht trivialer Kinematik verwendet. Bei Maschinen mit trivialer Kinematik (Fräsmaschinen, Drehmaschinen, Gantry-Typen) wird dieser Wert ignoriert. Note: Die Hexapod-Konfiguration von sim erfordert einen Wert ungleich Null für die Z-Koordinate.
- `TPMOD =` alternate_trajectory_planning Modul [tp_parms=Wert]
  Die `TPMOD`-Variable ist optional. Falls angegeben, verwenden Sie ein angegebenes (benutzerdefiniertes) Modul anstelle des Standardmoduls (tpmod). Modulparameter (tp_parms) können enthalten sein, wenn sie vom benannten Modul unterstützt werden. Die Einstellung kann über die Befehlszeile mit der Option -t ($ linuxcnc -h) überschrieben werden.
- `NO_PROBE_JOG_ERROR = 0` - Erlaubt die Umgehung der Prüfung, ob der Fühler ausgelöst hat, wenn Sie manuell joggen.
- `NO_PROBE_HOME_ERROR = 0` - Erlaubt die Umgehung der Prüfung, ob die Sonde ausgelöst wurde, während die Referenzfahrt läuft.

[KINS] Abschnitt

- `JOINTS = 3` - Gibt die Anzahl der Gelenke (Motoren) im System an. Eine Trivkins XYZ-Maschine mit einem Motor pro Achse hat beispielsweise 3 Gelenke. Eine Gantry-Maschine mit je einem Motor auf zwei Achsen und zwei Motoren auf der dritten Achse hat 4 Gelenke. (Diese Konfigurationsvariable kann von einer Benutzeroberfläche verwendet werden, um die Anzahl der Gelenke (num_joints) zu setzen, die dem Bewegungsmodul (motmod) angegeben wurde.)
- `KINEMATICS = trivkins` - Geben Sie ein Kinematikmodul für das Bewegungsmodul an. GUIs können diese Variable verwenden, um die `loadrt`-Zeile in HAL-Dateien für das motmod-Modul anzugeben. Weitere Informationen zu Kinematikmodulen finden Sie in der Manpage: `$ man kins`.

[AXIS_<Buchstabe>] Abschnitt

Der <*letter*> (engl. Buchstabe) gibt einen der folgenden Buchstaben an: X Y Z A B C U V W

- TYPE = LINEAR - (enum) The type of this axis, either LINEAR or ANGULAR. Required if this axis is not a default axis type. The default axis types are X,Y,Z,U,V,W = LINEAR and A,B,C = ANGULAR. X, Y and Z are always LINEAR. A LINEAR axis is a length: its words, offsets, stored positions and tool offsets follow G20 and G21 and [TRAJ]LINEAR_UNITS. An ANGULAR axis is an angle in degrees whatever G20 or G21 says, in [TRAJ]ANGULAR_UNITS for the machine, and may be a WRAPPED_ROTARY or have a LOCKING_INDEXER_JOINT. The feed of a move depends on the type too, see FEED_AXES in the [TRAJ] section. The preview, the DROs, touch off and jogging of the GUIs follow the type too.
- MAX_VELOCITY = 1.2 - (real) Maximum velocity for this axis in machine units per second.
- MAX_ACCELERATION = 20.0 - (real) Maximum acceleration for this axis in machine units per second squared.
- MAX_JERK = 0.0 - (real) Maximum jerk for this axis in machine units per second cubed. Used when S-curve trajectory planning is enabled. When set to 0 (default), no per-axis jerk limiting is applied.
- MIN_LIMIT = -1000 - (real) The minimum limit (soft limit) for axis motion, in machine units. When this limit is exceeded, the controller aborts axis motion. The axis must be homed before MIN_LIMIT is in force. For a rotary axis (A,B,C typ) with unlimited rotation having no MIN_LIMIT for that axis in the [AXIS_<letter>] section a value of -1e99 is used.
- MAX_LIMIT = 1000 - (real) The maximum limit (soft limit) for axis motion, in machine units. When this limit is exceeded, the controller aborts axis motion. The axis must be homed before MAX_LIMIT is in force. For a rotary axis (A,B,C typ) with unlimited rotation having no MAX_LIMIT for that axis in the [AXIS_<letter>] section a value of 1e99 is used.
- WRAPPED_ROTARY = 1 - (bool) When this is set to 1 for an ANGULAR axis the axis will move 0-359.999 degrees. Positive Numbers will move the axis in a positive direction and negative numbers will move the axis in the negative direction. It is ignored on a LINEAR axis. Mutually exclusive with ROTARY_MODULO (only one of the two can be set in a configuration).
- ROTARY_MODULO = 1 - (bool) Continuous-modulo rotary mode for an ANGULAR axis, ignored on a LINEAR one. Absolute moves on this axis take the shortest path: a G0 A0 from a current position of A=50000 moves only the residual angle (40 deg in this example), not 138 full revolutions. G-code may use absolute values of any magnitude (e.g. A=200000 for helical-spiral CAM output) without the +/-360 limit imposed by WRAPPED_ROTARY. The axis's current position parameter (#5423 to #5428, and the named equivalents #<_a> to #<_w> and #<_abs_a> to #<_abs_w>) reports position wrapped to the range 0 up to but not including 360 degrees; its probing parameter (#5064 to #5069) likewise. Internal commanded position and motion-side motion.traj.position/joint.<n>.pos-fb HAL pins remain accumulated to keep stepgens, encoders and PID loops consistent (no phantom unwind on wrap). The accumulated physical position can still be read into G-code from the HAL pin via the #<_hal[...]> syntax (see Numbered Parameters and the HAL_PIN_VARS setting in [RS274NGC] Section); for example #<_hal[joint.3.pos-fb]> returns the unwrapped joint position. Threading (G33, G33.1, G76) is rejected with an error if the modulo axis word is present in the block. Leave MIN_LIMIT and MAX_LIMIT unset for the axis and its joint: because the commanded position accumulates instead of wrapping, motion is refused with an "would exceed" error as soon as the accumulated position leaves the configured range, which on a bounded axis happens after the first few turns. A startup warning is printed if both limits are set to a bounded range. Mutually exclusive with WRAPPED_ROTARY: if both are set on the same axis a warning is printed at startup, ROTARY_MODULO is ignored and the axis keeps its WRAPPED_ROTARY behavior. Mid-program restart-from-line and DRO visual wrap in GUIs are not handled by this flag; these are left to the existing

restart machinery and to user-side HAL components if visual wrap is desired. Path control codes `M26` (shortest path, default) and `M27` (direction picked by the sign of the programmed value) are modal: once programmed they persist for all subsequent absolute moves on modulo axes until the other code is programmed or the program ends. For multi-turn moves, use `G91`.

- `LOCKING_INDEXER_JOINT = 4` - (int) This value selects a joint to use for a locking indexer for the specified axis <*letter*>. In this example, the joint is 4 which would correspond to the B axis for a XYZAB system with trivkins (identity) kinematics. When set, a G0 move for this axis will initiate an unlock with the `joint.4.unlock pin` then wait for the `joint.4.is-unlocked` pin then move the joint at the rapid rate for that joint. After the move the `joint.4.unlock` will be false and motion will wait for `joint.4.is-unlocked` to go false. Moving with other joints is not allowed when moving a locked rotary joint. It is ignored on a `LINEAR` axis. To create the unlock pins, use the motmod parameter:

  ```
  unlock_joints_mask=jointmask
  ```

  Die Bits der Jointmaske sind: (LSB)0:joint0, 1:joint1, 2:joint2, ...

  Beispiel: `loadrt motmod ... unlock_joints_mask=0x38` erzeugt Entsperrstifte für die Gelenke 3, 4, 5.

- `OFFSET_AV_RATIO = 0.1` - (real) If nonzero, this item enables the use of HAL input pins for external axis offsets:

  ```
  axis.<letter>.eoffset-enable
  axis.<letter>.eoffset-count
  axis.<letter>.eoffset-scale
  ```

Siehe das Kapitel: *External Axis Offsets* für Informationen zur Verwendung.

[JOINT_<num>] Abschnitte

Die <*num*> gibt die Gelenknummer 0 ... (num_joints-1) an. Der Wert von *num_joints* wird festgelegt durch `[KINS]JOINTS=`.

Die Abschnitte `[JOINT_0]`, `[JOINT_1]`, usw. enthalten allgemeine Parameter für die einzelnen Komponenten im Gelenksteuerungsmodul. Die Namen der Gelenkabschnitte beginnen bei 0 und reichen bis zur Anzahl der im Eintrag `[KINS]JOINTS` angegebenen Gelenke minus 1.

Typischerweise (bei Systemen, die *trivkins kinematics* verwenden, besteht eine 1:1-Entsprechung zwischen einem Gelenk und einem Achsenkoordinatenbuchstaben):

- JOINT_0 = X
- JOINT_1 = Y
- JOINT_2 = Z
- JOINT_3 = A
- JOINT_4 = B

- JOINT_5 = C
- JOINT_6 = U
- JOINT_7 = V
- JOINT_8 = W

Andere Kinematikmodule mit Identitätskinematik sind verfügbar, um Konfigurationen mit partiellen Achsensätzen zu unterstützen. Bei der Verwendung von trivkins mit `coordinates=XZ` sind die Beziehungen zwischen den Gelenkachsen beispielsweise wie folgt:

- JOINT_0 = X
- JOINT_1 = Z

Weitere Informationen über Kinematikmodule finden Sie in der Manpage *kins* (auf dem UNIX-Terminal geben Sie `man kins` ein).

- `TYPE = LINEAR` - (enum) The type of joint, either `LINEAR` or `ANGULAR`.
- `UNITS = INCH` - (enum) If specified, this setting overrides the related `[TRAJ] UNITS` setting, e.g., `[TRAJ]LINEAR_UNITS` if the `TYPE` of this joint is `LINEAR`, `[TRAJ]ANGULAR_UNITS` if the `TYPE` of this joint is `ANGULAR`.
- `MAX_VELOCITY = 1.2` - (real) Maximum velocity for this joint in machine units per second.
- `MAX_ACCELERATION = 20.0` - (real) Maximum acceleration for this joint in machine units per second squared.
- `MAX_JERK = 0.0` - (real) Maximum jerk for this joint in machine units per second cubed. Used when S-curve trajectory planning is enabled. When set to 0 (default), no per-joint jerk limiting is applied.
- `BACKLASH = 0.0000` - (real) Backlash in machine units. Backlash compensation value can be used to make up for small deficiencies in the hardware used to drive an joint. If backlash is added to an joint and you are using steppers the `STEPGEN_MAXACCEL` must be increased to 1.5 to 2 times the `MAX_ACCELERATION` for the joint. Excessive backlash compensation can cause an joint to jerk as it changes direction. If a COMP_FILE is specified for a joint BACKLASH is not used.
- `COMP_FILE =` *file.extension* - (string) The compensation file consists of map of position information for the joint. Compensation file values are in machine units. Each set of values are are on one line separated by a space. The first value is the nominal value (the commanded position). The second and third values depend on the setting of `COMP_FILE_TYPE`. Points in between nominal values are interpolated between the two nominals. Compensation files must start with the smallest nominal and be in ascending order to the largest value of nominals. File names are case sensitive and can contain letters and/or numbers. Currently the limit inside LinuxCNC is for 256 triplets per joint.

  Wenn `COMP_FILE` für ein Gelenk angegeben ist, wird `BACKLASH` nicht verwendet.

- `COMP_FILE_TYPE = 0` or `1` - (int) Specifies the type of compensation file. The first value is the nominal (commanded) position for both types.
  A `COMP_FILE_TYPE` must be specified for each `COMP_FILE`.
  - *Typ 0:* Der zweite Wert gibt die Ist-Position bei Bewegung des Gelenks in positiver Richtung an (steigender Wert). Der dritte Wert gibt die Ist-Position bei Bewegung des Gelenks in negativer

Richtung an (fallender Wert).

Typ 0 Beispiel

```
-1.000 -1.005 -0.995
0.000 0.002 -0.003
1.000 1.003 0.998
```

- *Typ 1:* Der zweite Wert gibt die positive Abweichung vom Sollwert bei Fahrt in positiver Richtung an. Der dritte Wert gibt die negative Abweichung vom Sollwert an, während die Fahrt in negativer Richtung erfolgt.

Typ 1 Beispiel

```
-1.000 0.005 -0.005
0.000 0.002 -0.003
1.000 0.003 -0.004
```

- `MIN_LIMIT = -1000` - (real) The minimum limit for joint motion, in machine units. When this limit is reached, the controller aborts joint motion. For a rotary joint with unlimited rotation having no `MIN_LIMIT` for that joint in the `[JOINT_N]` section a the value -1e99 is used.
- `MAX_LIMIT = 1000` - (real) The maximum limit for joint motion, in machine units. When this limit is reached, the controller aborts joint motion. For a rotary joint with unlimited rotation having no `MAX_LIMIT` for that joint in the `[JOINT_N]` section a the value 1e99 is used.

NOTE Für **Identitäts**-Kinematiken müssen die Einstellungen `[JOINT_N]MIN_LIMIT` /`MAX_LIMIT` den entsprechenden (eins-zu-eins-identischen) `[AXIS_L]`-Grenzwerten entsprechen oder diese überschreiten. Diese Einstellungen werden beim Starten überprüft, wenn die trivkins-Kinematikmodule angegeben werden.

NOTE Die Einstellungen `[JOINT_N]MIN_LIMIT`/`MAX_LIMIT` werden beim Joggen im Gelenkmodus vor der Referenzfahrt erzwungen. Nach der Referenzfahrt werden die Koordinatengrenzen `[AXIS_L]MIN_LIMIT`/`MAX_LIMIT` als Beschränkungen für die Achsenbewegung (Koordinatenbuchstaben) und für die Bahnplanung bei G-Code-Bewegungen (Programme und MDI-Befehle) verwendet. Der Trajektorienplaner arbeitet im kartesischen Raum (XYZABCUVW) und hat keine Informationen über die Bewegung von Gelenken, die von **jedem** Kinematikmodul implementiert werden. Es ist möglich, dass bei G-Code, der die Positionsgrenzen der Trajektorienplanung einhält, Verletzungen der Gelenkgrenzen auftreten, wenn nicht identische Kinematiken verwendet werden. Das Bewegungsmodul erkennt immer Verletzungen der Gelenkpositionsgrenzen und Fehler, wenn sie während der Ausführung von G-Code-Befehlen auftreten. Siehe dazu auch GitHub issue #97.

- `MIN_FERROR = 0.010` - (real) This is the value in machine units by which the joint is permitted to deviate from commanded position at very low speeds. If MIN_FERROR is smaller than FERROR, the two produce a ramp of error trip points. You could think of this as a graph where one dimension is speed and the other is permitted following error. As speed increases the amount of following error also increases toward the `FERROR` value.

- FERROR = 1.0 - (real) FERROR is the maximum allowable following error, in machine units. If the difference between commanded and sensed position exceeds this amount, the controller disables servo calculations, sets all the outputs to 0.0, and disables the amplifiers. If MIN_FERROR is present in the INI file, velocity-proportional following errors are used. Here, the maximum allowable following error is proportional to the speed, with FERROR applying to the rapid rate set by [TRAJ]MAX_VELOCITY, and proportionally smaller following errors for slower speeds. The maximum allowable following error will always be greater than MIN_FERROR. This prevents small following errors for stationary axes from inadvertently aborting motion. Small following errors will always be present due to vibration, etc.
- LOCKING_INDEXER = 1 - (bool) Indicates the joint is used as a locking indexer.

Referenzfahrt (engl. homing)

Diese Parameter beziehen sich auf die Ausführung der Referenzfahrt (engl. homing), für eine bessere Erklärung lesen Sie das Referenzfahrt-Konfiguration Kapitel.

- HOME = 0.0 - (real) The position that the joint will go to upon completion of the homing sequence.
- HOME_OFFSET = 0.0 - (real) The joint position of the home switch or index pulse, in machine units. When the home point is found during the homing process, this is the position that is assigned to that point. When sharing home and limit switches and using a home sequence that will leave the home/limit switch in the toggled state, the home offset can be used define the home switch position to be other than 0 if your HOME position is desired to be 0.
- HOME_SEARCH_VEL = 0.0 - (real) Initial homing velocity in machine units per second. Sign denotes direction of travel. A value of zero means assume that the current location is the home position for the machine. If your machine has no home switches you will want to leave this value at zero.
- HOME_LATCH_VEL = 0.0 - (real) Homing velocity in machine units per second to the home switch latch position. Sign denotes direction of travel.
- HOME_FINAL_VEL = 0.0 - (real) Velocity in machine units per second from home latch position to home position. If left at 0 or not included in the joint rapid velocity is used. Must be a positive number.
- HOME_SEARCH_DIST = 0.0 - (real) Farthest the home search move may travel in machine units without finding the switch. Past it the joint stops and the homing fails. Zero (the default) leaves the move bounded only by twice the joint's travel.
- HOME_LATCH_DIST = 0.0 - (real) Farthest the back-off, latch and index search moves may each travel in machine units without result. Past it the joint moves back to where that move began and the homing fails. Zero (the default) leaves those moves bounded only by twice the joint's travel.
- HOME_USE_INDEX = NO - (bool) If the encoder used for this joint has an index pulse, and the motion card has provision for this signal you may set it to yes. When it is yes, it will affect the kind of home pattern used. Currently, you can't home to index with steppers unless you're using StepGen in velocity mode and PID.
- HOME_INDEX_NO_ENCODER_RESET = NO - (bool) Use YES if the encoder used for this joint does not reset its counter when an index pulse is detected after assertion of the joint index_enable HAL pin. Applicable only for HOME_USE_INDEX = YES.
- HOME_IGNORE_LIMITS = NO - (bool) When you use the limit switch as a home switch and the limit

switch this should be set to YES. When set to YES the limit switch for this joint is ignored when homing. You must configure your homing so that at the end of your home move the home/limit switch is not in the toggled state you will get a limit switch error after the home move.

- `HOME_IS_SHARED = NO` - (bool) If the home input is shared by more than one joint set to true to prevent homing from starting if the one of the shared switches is already closed. Set to false (the default) to permit homing if a switch is closed.
- `HOME_ABSOLUTE_ENCODER = 0 | 1 | 2` - (int) Used to indicate the joint uses an absolute encoder. At a request for homing, the current joint value is set to the `HOME_OFFSET` value. If the `HOME_ABSOLUTE_ENCODER` setting is 1, the machine makes the usual final move to the `HOME` value. If the `HOME_ABSOLUTE_ENCODER` setting is 2, no final move is made.
- `HOME_SEQUENCE =` <*n*> - (int) Used to define the "Home All" sequence. <*n*> must start at `0` or `1` or `-1`. Additional sequences may be specified with numbers increasing by 1 (in absolute value). Skipping of sequence numbers is not allowed. If a `HOME_SEQUENCE` is omitted, the joint will not be homed by the "Home All" function. More than one joint can be homed at the same time by specifying the same sequence number for more than one joint. A negative sequence number is used to defer the final move for all joints having that (negative or positive) sequence number. For additional info, see: HOME SEQUENCE.
- `VOLATILE_HOME = 0` - (bool) When enabled (set to `1`) this joint will be unhomed if the Machine Power is off or if E-Stop is on. This is useful if your machine has home switches and does not have position feedback such as a step and direction driven machine.

Servos

Diese Parameter sind relevant für Gelenke, die von Servos gesteuert werden.

WARNING

Das Folgende sind benutzerdefinierte INI-Datei-Einträge, die Sie in einer Beispiel-INI-Datei oder einer vom Assistenten generierten Datei finden können. Diese werden nicht von der LinuxCNC-Software verwendet. Sie sind nur dazu da, alle Einstellungen an einem Ort zu speichern. Für weitere Informationen über benutzerdefinierte INI-Datei-Einträge siehe den Unterabschnitt Benutzerdefiniert Abschnitte und Variablen.

Die folgenden Elemente können von einer PID-Komponente verwendet werden, wobei davon ausgegangen wird, dass die Ausgabe in Volt erfolgt.

- `DEADBAND = 0.000015` - (real) How close is close enough to consider the motor in position, in machine units.

  Dies wird oft auf einen Abstand eingestellt, der 1, 1,5, 2 oder 3 Encoderzählungen entspricht, aber es gibt keine strengen Regeln. Lockere (größere) Einstellungen ermöglichen ein geringeres Hunting' des Servos auf Kosten einer geringeren Genauigkeit. Engere (kleinere) Einstellungen versuchen eine höhere Genauigkeit auf Kosten von mehr Servo *Hunting*. Ist es wirklich genauer, wenn es auch unsicherer ist? Generell ist es gut, das Hunting' der Servos zu vermeiden oder zumindest zu begrenzen, wenn Sie können.

  Seien Sie vorsichtig, wenn Sie unter 1 Geberzahl gehen, da Sie einen Zustand schaffen können, in

dem Ihr Servo an keiner Stelle zufrieden ist. Dies kann über *Hunting* (langsam) bis hin zu *Nervös* (schnell) und sogar zu *Quietschen* gehen, was leicht mit Oszillation, verursacht durch unsachgemäße Abstimmung, verwechselt werden kann. Es ist besser, anfangs ein oder zwei Zählzeiten weniger zu spielen, zumindest bis man die erste *Grobabstimmung* hinter sich hat.

Beispiel für die Berechnung von Maschineneinheiten pro Encoderimpuls zur Bestimmung des `DEADBAND`-Wertes:

$$\frac{1\,revolution}{1000\,lines} X \frac{1\,line}{4\,pulse/line} X \frac{0.2\,units}{1\,revolution} = \frac{0.200\,units}{4000\,pulses} = \frac{0.00005\,units}{1\,pulse}$$

- `BIAS = 0.000` - Dies wird von hm2-servo und einigen anderen verwendet. Bias ist ein konstanter Betrag, der zum Ausgang addiert wird. In den meisten Fällen sollte er auf Null belassen werden. Er kann jedoch manchmal nützlich sein, um Offsets in Servoverstärkern zu kompensieren oder das Gewicht eines Objekts auszugleichen, das sich vertikal bewegt. Der Bias (auch Vorspannung) wird ausgeschaltet, wenn die PID-Schleife deaktiviert ist, genau wie alle anderen Komponenten des Ausgangs.
- `P = 50` - Die proportionale Verstärkung für das Gelenkservo. Dieser Wert multipliziert den Fehler zwischen befohlener und tatsächlicher Position in Maschineneinheiten, was zu einem Beitrag zur berechneten Spannung für den Motorverstärker führt. Die Einheiten für die P-Verstärkung sind Volt pro Maschineneinheit, z. B. $\frac{volts}{unit}$
- `I = 0` - Die integrale Verstärkung für das Gelenkservo. Der Wert multipliziert den kumulativen Fehler zwischen befohlener und tatsächlicher Position in Maschineneinheiten, was zu einem Beitrag zur berechneten Spannung für den Motorverstärker führt. Die Einheiten für die I-Verstärkung sind Volt pro Maschineneinheit pro Sekunde, z. B. $\frac{volts}{unit\,second}$
- `D = 0` - Die Ableitungsverstärkung für das Gelenkservo. Der Wert multipliziert die Differenz zwischen dem aktuellen und dem vorherigen Fehler, was zu einem Beitrag zur berechneten Spannung für den Motorverstärker führt. Die Einheiten für die D-Verstärkung sind Volt pro Maschineneinheit pro Sekunde, z. B. $\frac{volts}{unit\,second}$
- `FF0 = 0` - Die Vorwärtsverstärkung 0ter Ordnung. Diese Zahl wird mit der befohlenen Position multipliziert, was zu einem Beitrag zur berechneten Spannung für den Motorverstärker führt. Die Einheiten für die FF0-Verstärkung sind Volt pro Maschineneinheit, z. B. $\frac{volts}{unit}$
- `FF1 = 0` - Die Vorwärtsverstärkung erster Ordnung. Diese Zahl wird mit der Änderung der befohlenen Position pro Sekunde multipliziert, was zu einem Beitrag zur berechneten Spannung für den Motorverstärker führt. Die Einheiten für die FF1-Verstärkung sind Volt pro Maschineneinheit pro Sekunde, z. B. $\frac{volts}{unit\,second}$
- `FF2 = 0` - Die Vorwärtsverstärkung zweiter Ordnung. Diese Zahl wird mit der Änderung der befohlenen Position pro Sekunde multipliziert, was zu einem Beitrag zur berechneten Spannung für den Motorverstärker führt. Die Einheiten für die FF2-Verstärkung sind Volt pro Maschineneinheit pro Sekunde, z. B. $\frac{volts}{unit\,second^2}$

- `OUTPUT_SCALE = 1.000`
- `OUTPUT_OFFSET = 0.000`

Diese beiden Werte sind die Skalierungs- und Offset-Faktoren für den gemeinsamen Ausgang zu den Motorverstärkern.

Der zweite Wert (Offset) wird vom berechneten Ausgang (in Volt) subtrahiert und durch den ersten Wert (Skalierungsfaktor) geteilt, bevor er in die D/A-Wandler geschrieben wird. Die Einheiten für den Skalenwert sind in echten Volt pro DAC-Ausgangsspannung. Die Einheiten für den Offset-Wert sind in Volt. Diese können zur Linearisierung eines DAC verwendet werden. Insbesondere beim Schreiben von Ausgängen, die LinuxCNC erste wandelt die gewünschte Ausgabe in Quasi-SI-Einheiten zu rohen Aktor Werte, z. B. Volt für einen Verstärker DAC. Diese Skalierung sieht wie folgt aus: $raw = \frac{output - offset}{scale}$

Der Wert für die Skalierung kann analytisch ermittelt werden, indem eine Einheitenanalyse durchgeführt wird, d. h. die Einheiten sind [Ausgangs-SI-Einheiten]/[Aktuatoreinheiten]. Beispiel: Bei einer Maschine mit einem Verstärker im Geschwindigkeitsmodus ergibt 1 V eine Geschwindigkeit von 250 mm/s.

$$amplifier[volts] = (output[\frac{mm}{sec}] - offset[\frac{mm}{sec}]) / 250 \frac{mm}{secvolt}$$

Beachten Sie, dass die Einheiten des Offsets in Maschineneinheiten angegeben sind, z. B. mm/s, und dass sie von den Sensormesswerten abgezogen werden. Den Wert für diesen Offset erhalten Sie, indem Sie den Wert Ihres Ausgangs finden, der 0,0 für den Aktor-/Stellgliedausgang ergibt. Bei einem linearisierten DAC ist dieser Offset normalerweise 0,0.

Skalierung und Offset können auch zur Linearisierung des DAC verwendet werden. Diese Werte spigeln dann die kombinierten Auswirkungen von Verstärkung, Nicht-Linearität des DAC, DAC-Einheiten usw. wider.

Gehen Sie dazu folgendermaßen vor.

1. Erstellen Sie eine Kalibrierungstabelle für den Ausgang, indem Sie den DAC mit einer gewünschten Spannung betreiben und das Ergebnis messen.
2. Führen Sie eine lineare Anpassung nach dem Prinzip der kleinsten Quadrate durch, um die Koeffizienten a und b so zu ermitteln, dass $measured = a * raw + b$
3. Beachten Sie, dass wir eine Rohausgabe wünschen, bei der das gemessene Ergebnis mit der befohlenen Ausgabe identisch ist. Das bedeutet
   a. $command = a * raw + b$
   b. $raw = (command - b) / a$
4. Folglich können die Koeffizienten a und b aus der linearen Anpassung direkt als Skala und Offset für den Regler verwendet werden.

In der folgenden Tabelle finden Sie ein Beispiel für Spannungsmessungen.

Table 4. Messungen der Ausgangsspannung

Roh	Gemessen
-10	-9.93
-9	-8.83
0	-0.03
1	0.96
9	9.87
10	10.87

- `MAX_OUTPUT = 10` - Der maximale Wert für den Ausgang der PID-Kompensation, der in den Motorverstärker geschrieben wird, in Volt. Der berechnete Ausgangswert wird auf diesen Grenzwert geklemmt. Der Grenzwert wird vor der Skalierung auf rohe Ausgabeeinheiten angewendet. Der Wert wird symmetrisch sowohl auf die Plus- als auch auf die Minusseite angewandt.
- `INPUT_SCALE = 20000` - in Beispielkonfigurationen
- `ENCODER_SCALE = 20000` - in PnCconf erstellten Konfigurationen

Gibt die Anzahl der Impulse an, die einer Bewegung um eine Maschineneinheit entspricht, wie im Abschnitt `[TRAJ]` eingestellt. Bei einem linearen Gelenk entspricht eine Maschineneinheit der Einstellung von `LINEAR_UNITS`. Für ein Winkelgelenk entspricht eine Einheit der Einstellung in `ANGULAR_UNITS`. Eine zweite Zahl, falls angegeben, wird ignoriert. Bei einem Drehgeber mit 2000 Umdrehungen pro Minute, einem Getriebe mit 10 Umdrehungen pro Zoll und den gewünschten Einheiten in Zoll ergibt sich zum Beispiel Folgendes:

$$input\,scale = 2000\,\frac{counts}{rev} * 10\,\frac{rev}{inch} = 20000\,\frac{counts}{inch}$$

Schrittmotoren

Diese Parameter sind relevant für Gelenke, die von Schrittmotoren gesteuert werden.

WARNING

Das Folgende sind benutzerdefinierte INI-Datei-Einträge, die Sie in einer Beispiel-INI-Datei oder einer vom Assistenten generierten Datei finden können. Diese werden nicht von der LinuxCNC-Software verwendet und sind nur dazu gedacht, alle Einstellungen an einem Ort zu platzieren. Weitere Informationen über benutzerdefinierte INI-Datei-Einträge finden Sie im Unterabschnitt Custom Sections and Variables.

Die folgenden Elemente können von einer Schrittgenerator (engl. StepGen)-Komponente verwendet werden.

- `SCALE = 4000` - in Beispielkonfigurationen
- `STEP_SCALE = 4000` - in PnCconf erstellten Konfigurationen

Gibt die Anzahl der Impulse an, die einer Bewegung einer Maschineneinheit entspricht, wie im Abschnitt `[TRAJ]` eingestellt. Bei Schrittmotor-(engl. stepper)-systemen ist dies die Anzahl der

Schrittimpulse, die pro Maschineneinheit ausgegeben werden. Bei einem Lineargelenk entspricht eine Maschineneinheit der Einstellung von LINEAR_UNITS. Für ein Winkelgelenk entspricht eine Einheit der Einstellung in ANGULAR_UNITS. Bei Servosystemen ist dies die Anzahl der Rückmeldeimpulse pro Maschineneinheit. Eine zweite Zahl, falls angegeben, wird ignoriert.

Bei einem 1,8-Grad-Schrittmotor, der mit halben Schritten bewegt (engl. half-stepping) wird, und einem Getriebe mit 10 Umdrehungen pro Zoll und gewünschten Maschineneinheiten in Zoll ergibt sich beispielsweise Folgendes:

$$input\,scale = \frac{2\,steps}{1.8\,degrees} * 360\,\frac{degree}{rev} * 10\,\frac{rev}{inch} = 4000\,\frac{steps}{inch}$$

NOTE Alte INI und HAL Dateien verwendeten INPUT_SCALE für diesen Wert.

- ENCODER_SCALE = 20000 (wird optional in PnCconf-Konfigurationen verwendet) - Gibt die Anzahl der Impulse an, die einer Bewegung um eine Maschineneinheit entspricht, wie im Abschnitt [TRAJ] festgelegt. Bei einem linearen Gelenk entspricht eine Maschineneinheit der Einstellung von LINEAR_UNITS. Für ein Winkelgelenk entspricht eine Einheit der Einstellung in ANGULAR_UNITS. Eine zweite Zahl, falls angegeben, wird ignoriert. Bei einem Drehgeber mit 2000 Umdrehungen pro Minute, einem Getriebe mit 10 Umdrehungen pro Zoll und den gewünschten Einheiten in Zoll ergibt sich zum Beispiel Folgendes:

$$input\,scale = 2000\,\frac{counts}{rev} * 10\,\frac{rev}{inch} = 20000\,\frac{counts}{inch}$$

- STEPGEN_MAXACCEL = 21.0 - Beschleunigungsgrenze für den Schrittgenerator. Dieser Wert sollte 1% bis 10% größer sein als die gemeinsame MAX_ACCELERATION. Dieser Wert verbessert die Abstimmung des StepGen's "Positionsschleife". Wenn Sie einem Gelenk eine Spielkompensation hinzugefügt haben, sollte dieser Wert 1,5 bis 2 Mal größer als MAX_ACCELERATION sein.
- STEPGEN_MAXVEL = 1.4 - Ältere Konfigurationsdateien haben auch eine Geschwindigkeitsgrenze für den Schrittgenerator. Falls angegeben, sollte sie ebenfalls 1 % bis 10 % größer sein als die gemeinsame MAX_VELOCITY. Nachfolgende Tests haben gezeigt, dass die Verwendung von STEPGEN_MAXVEL die Abstimmung der Positionsschleife von StepGen nicht verbessert.

[SPINDLE_<num>] Abschnitt(e)

Die <*num*> spezifiziert die Spindelnummer 0 ... (num_spindles-1)
Der Wert von *num_spindles* wird durch [TRAJ]SPINDLES= gesetzt.
Standardmäßig beträgt die maximale Geschwindigkeit der Spindel nach vorne und hinten ca. 2147483000 RPM.
Standardmäßig ist die Mindestgeschwindigkeit der Spindel nach vorne und umgekehrt 0 RPM.
Standardmäßig beträgt das Inkrement 100 RPM.
Sie ändern diese Standardeinstellung, indem Sie die folgenden INI-Variablen festlegen:

NOTE Diese Einstellungen sind für die Bewegungssteuerung (engl. motion controler) verantwortlich Komponente. Kontrollschirme können diese Einstellungen weiter begrenzen.

- `MAX_FORWARD_VELOCITY = 20000` - (real) The maximum spindle speed (in rpm) for the specified spindle. Optional. This will also set MAX_REVERSE_VELOCITY to the negative value unless overridden.
- `MIN_FORWARD_VELOCITY = 3000` - (real) The minimum spindle speed (in rpm) for the specified spindle. Optional. Many spindles have a minimum speed below which they should not be run. Any spindle speed command below this limit will be /increased/ to this limit.
- `MAX_REVERSE_VELOCITY = 20000` - (real) This setting will default to `MAX_FORWARD_VELOCITY` if omitted. It can be used in cases where the spindle speed is limited in reverse. Set to zero for spindles which must not be run in reverse. In this context "max" refers to the absolute magnitude of the spindle speed.
- `MIN_REVERSE_VELOCITY = 3000` - (real) This setting is equivalent to `MIN_FORWARD_VELOCITY` but for reverse spindle rotation. It will default to the MIN_FORWARD_VELOCITY if omitted.
- `INCREMENT = 200` - (real) Sets the step size for spindle speed increment / decrement commands. This can have a different value for each spindle. This setting is effective with AXIS and Touchy but note that some control screens may handle things differently.
- `HOME_SEARCH_VELOCITY = 100` - (real) FIXME: Spindle homing not yet working. Sets the homing speed (rpm) for the spindle. The spindle will rotate at this velocity during the homing sequence until the spindle index is located, at which point the spindle position will be set to zero. Note that it makes no sense for the spindle home position to be any value other than zero, and so there is no provision to do so.
- `HOME_SEQUENCE = 0` - (int) FIXME: Spindle homing not yet working Controls where in the general homing sequence the spindle homing rotations occur. Set the `HOME_SEARCH_VELOCITY` to zero to avoid spindle rotation during the homing sequence.

[EMCIO] Abschnitt

- `TOOL_TABLE = tool.tbl` - (string) The file which contains tool information, described in the User Manual.
- `DB_PROGRAM = db_program` - (string) Path to an executable program that manages tool data. When a DB_PROGRAM is specified, a TOOL_TABLE entry is ignored.
- `TOOL_CHANGE_POSITION = 0 0 2` - Gibt die XYZ-Position an, die bei einem Werkzeugwechsel angefahren wird, wenn drei Ziffern verwendet werden. Gibt die XYZABC-Position an, wenn 6 Ziffern verwendet werden. Gibt die XYZABCUVW-Position an, wenn 9 Ziffern verwendet werden. Werkzeugwechsel können kombiniert werden. Wenn Sie z. B. die Pinole nach oben mit der Wechselposition kombinieren, können Sie zuerst die Z-Position und dann die X- und Y-Position verschieben.
- `TOOL_CHANGE_WITH_SPINDLE_ON = 1` - (bool) The spindle will be left on during the tool change when the value is 1. Useful for lathes or machines where the material is in the spindle, not the tool.
- `TOOL_CHANGE_QUILL_UP = 1` - (bool) The Z axis will be moved to machine zero prior to the tool change when the value is 1. This is the same as issuing a `G0 G53 Z0`.
- `TOOL_CHANGE_AT_G30 = 1` - (bool) The machine is moved to reference point defined by parameters 5181-5186 for G30 if the value is 1. For more information see G-code Parameters and G-code G30-G30.1.

- `RANDOM_TOOLCHANGER = 1` - (bool) This is for machines that cannot place the tool back into the pocket it came from. For example, machines that exchange the tool in the active pocket with the tool in the spindle.

[MTCONNECT] Section

Optional. Configures the `mtconnect-agent`(1) userspace component, which publishes machine status, kinematics and tool data using the MTConnect standard. Load it from a HAL file with `loadusr -W mtconnect-agent`. The MTConnect device model is generated automatically from the `[TRAJ]`, `[KINS]`, `[AXIS_<letter>]` and `[JOINT_<num>]` sections, so only the keys below are required.

- `ENABLE = 1` - (bool) Enable the agent. Default 1.
- `DEVICE_NAME = my_mill` - (string) MTConnect device name. Default: `[EMC]MACHINE`.
- `UUID = 0e2c...` - (string) MTConnect device uuid.
- `TRANSPORT = http` - (string) Comma-separated list of transports: `http`, `mqtt`, `shdr`. Default `http`.
- `HTTP_PORT = 5000` - (int) Port for the embedded HTTP agent.
- `HTTP_BIND = 127.0.0.1` - (string) Interface to bind the HTTP agent to. Default 127.0.0.1 (loopback only); set to `0.0.0.0` to expose on the network.
- `SAMPLE_HZ = 10` - (real) Poll and publish rate in Hz.
- `MQTT_BROKER`, `MQTT_PORT`, `MQTT_PREFIX`, `MQTT_USERNAME`, `MQTT_PASSWORD` - Settings for the standard MTConnect MQTT binding. Requires the `python3-paho-mqtt` package.

The embedded HTTP agent serves `/probe`, `/current`, `/sample` and `/assets`, and referenced geometry under `/models/`. The spindle speed band from `[SPINDLE_0]MIN_FORWARD_VELOCITY` /`MAX_FORWARD_VELOCITY` is advertised as an MTConnect Specification. See `mtconnect-agent`(1) for the full reference.

4.5. MTConnect

4.5.1. What is MTConnect?

MTConnect is an open, read-only standard for getting data out of manufacturing equipment. A machine (or an **agent** in front of it) publishes a structured, XML-based description of itself and a live stream of its state, which any compliant software can read: shop dashboards, machine monitoring and OEE tools, digital twins, and CAD/CAM systems. It is read-only by design - nothing can command the machine through it - which makes it safe to expose.

An MTConnect agent answers a few standard requests:

- **probe** - the device model: the machine's components, axes, kinematics and the list of data items it reports.
- **current** - the latest value of every data item.
- **sample** - a history of values by sequence number.

- **assets** - richer objects such as cutting tools.

LinuxCNC provides a native agent, `mtconnect-agent`(1), so no third-party software is required.

4.5.2. Enabling the agent

Add a `[MTCONNECT]` section to the INI (see [MTCONNECT] Section for all keys) and load the agent from a HAL file:

INI-Datei

```
[MTCONNECT]
ENABLE      = 1
DEVICE_NAME = my_mill
TRANSPORT   = http
HTTP_PORT   = 5000
```

HAL-Datei

```
loadusr -W mtconnect-agent
```

`-W` waits until the component is ready before HAL processing continues. The INI is taken from the `INI_FILE_NAME` environment variable, so no argument is needed. The MTConnect device model - including the kinematic description - is generated automatically from `[TRAJ]`, `[KINS]`, `[AXIS_<letter>]` and `[JOINT_<num>]`.

By default the HTTP server listens on `127.0.0.1` (loopback) only. To let other machines on the network read it, set `HTTP_BIND = 0.0.0.0`.

Once running:

```
curl http://localhost:5000/probe
curl http://localhost:5000/current
curl http://localhost:5000/assets
```

4.5.3. HAL pins

Loaded from a HAL file, the component exposes pins you can link:

- `mtconnect-agent.enable` (bool, in) - gate polling and publishing.
- `mtconnect-agent.active` (bool, out) - TRUE while polling and serving.
- `mtconnect-agent.connected` (bool, out) - TRUE while the MQTT broker link is up.
- `mtconnect-agent.heartbeat` (uint, out) - increments each poll.

4.5.4. Exposing custom HAL pins

A lot of useful machine data lives in HAL pins that are not part of `linuxcnc.stat()` - spindle load, drive or board temperatures, air pressure, vacuum, custom sensors. Any HAL pin, signal or parameter

can be published as an MTConnect data item with a HAL_ITEM line; the agent reads it each poll with hal.get_value (no HAL-file wiring needed) and it then appears in /probe, /current, /sample and over MQTT and SHDR like any other item.

```
[MTCONNECT]
HAL_ITEM = pin=spindle.0.load, id=spindle_load, type=LOAD, units=PERCENT,
component=spindle
HAL_ITEM = pin=hm2_5i25.temp,  id=board_temp,   type=TEMPERATURE, units=CELSIUS
```

Fields: pin= (the HAL name), id= (the data item id, also its MQTT/SHDR key), type= (a standard MTConnect SAMPLE type), and optionally units=, name=, subType=, and component=. component= chooses the host: the default sensors (a generic Sensor component), or an existing one - spindle, controller, path, or an axis letter.

Only standard MTConnect SAMPLE types are accepted (for example LOAD, TEMPERATURE, PRESSURE, VOLTAGE, AMPERAGE, WATTAGE, FREQUENCY, ANGLE, VELOCITY, ACCELERATION, TORQUE, FILL_LEVEL, FLOW). A custom type would need the LinuxCNC extension schema and cannot be represented in the base MTConnectStreams schema, so it is rejected with a warning listing the supported types; the agent still starts.

4.5.5. The kinematic description

Standard MTConnect models each axis as a Linear or Rotary component with a Motion element (PRISMATIC/REVOLUTE plus a direction vector), and describes frames with CoordinateSystems. Travel limits are published as Specifications.

LinuxCNC specifics that have no standard representation - the kinematics module name, the coordinates= string, and the joint-to-axis map - are carried in a compact extension block, x:Kinematics, in the LinuxCNC extension namespace urn:linuxcnc:mtconnect:1. This is the primary contract for a consumer (for example a FreeCAD machine-configuration plugin) that wants to reconstruct the machine automatically.

4.5.6. Solid models (geometry)

The device model can carry 3D geometry so a standard MTConnect twin viewer (for example the one at demo.mtconnect.org, or TrakHound) can render the machine. Geometry is **referenced, never streamed**: the /probe document points to mesh files with SolidModel elements, and the agent serves each mesh at /models/<name>. A viewer fetches each mesh once and animates it from /current using the CoordinateSystems and Motion chain in /probe. This is standard MTConnect - LinuxCNC bundles no viewer and no JavaScript.

With MODEL_AUTO = 1 the agent generates simple placeholder box geometry from the travel limits, so any machine gets usable geometry with no mesh files. Supply STL/OBJ/glTF meshes (authored in millimetres) with MODEL_BASE, MODEL_X, etc. for higher fidelity. Because a trivial-kinematics INI does not describe the mechanical assembly, MODEL_CHAIN, MODEL_PARENT_<axis> and MODEL_INVERT describe the nesting, branching (e.g. a quill that carries the tool) and which links physically move opposite the reported tool-relative coordinate.

4.5.7. MQTT

With `mqtt` in the `TRANSPORT` list and the `python3-paho-mqtt` package installed, the agent also publishes to the standard MTConnect MQTT topics (`<prefix>/Probe/<uuid>` retained, plus `Current`, `Sample` and `Asset`). A consumer discovers the whole device from the retained Probe topic.

Home Assistant (optional)

Home Assistant's MQTT Discovery is an HA-specific convention, not an open standard, so it is deliberately not part of the core agent. An optional bridge, `mtconnect-ha-bridge` (in the MTConnect example config's `contrib/` directory), reuses the agent's device model to publish Home Assistant discovery so HA auto-creates a device with one sensor per value - no YAML required. It is configured entirely with command-line arguments, so no Home Assistant setting is placed in the machine INI:

```
loadusr -W mtconnect-ha-bridge \
  --broker=[HA]BROKER --username=[HA]USER --password=[HA]PASSWORD
```

4.5.8. Using an external MTConnect agent (SHDR)

Instead of (or in addition to) the embedded agent, LinuxCNC can act as an **SHDR adapter** and feed a standard external MTConnect agent such as the reference cppagent. SHDR is the MTConnect adapter protocol: a line-oriented TCP stream the agent connects to and reads. Add `shdr` to the `TRANSPORT` list; the adapter listens on `SHDR_PORT` (default 7878):

```
[MTCONNECT]
TRANSPORT = http, shdr
SHDR_PORT = 7878
```

The device **model** is not sent over SHDR - only data values, as `<timestamp>|<dataItemId>|<value>` lines. Configure the external agent with a `Devices.xml`, which the generated probe document doubles as:

```
mtconnect-agent --dump-probe my_machine.ini > Devices.xml
```

The `dataItemId` values in the SHDR stream match the ids in that `Devices.xml`. (The SHDR port is reachable from other hosts by design, since an external agent usually runs elsewhere.)

4.5.9. The LinuxCNC extension schema

A few data items have no standard MTConnect type: coolant flood/mist state and the active XY work-coordinate rotation (`G10 L2 R`). These are published in the `urn:linuxcnc:mtconnect:1` namespace and defined by the extension schema `mtconnect-linuxcnc-1.xsd`, installed under `share/linuxcnc/mtconnect` and served by the agent at `/mtconnect-linuxcnc-1.xsd`. The streaming documents validate against the official MTConnect schemas together with this extension schema.

4.6. Konfiguration der Referenzfahrt (engl. homing)

4.6.1. Übersicht

Die Referenzfahrt legt den Nullpunkt der G53-Maschinenkoordinaten fest. Softlimits werden relativ zum Maschinenursprung definiert. Eine korrekt konfigurierte und funktionierende Maschine bewegt sich nicht über die Soft(ware)-Grenzen hinaus und der Maschinenursprung ist so wiederholbar eingestellt wie der Referenzschalter/Indexmechanismus. Linuxcnc kann mit dem Auge (Ausrichtungsmarken), mit Schaltern, mit Schaltern und einem Encoder-Index oder mit Absolut-Encodern ausgerichtet werden. Homing scheint einfach genug - bewegen Sie einfach jedes Gelenk zu einer bekannten Position, und stellen Sie LinuxCNC's interne Variablen entsprechend. Allerdings haben verschiedene Maschinen unterschiedliche Anforderungen, und Homing ist eigentlich ziemlich kompliziert.

NOTE Es ist zwar möglich, LinuxCNC ohne Referenzschalter/Referenzfahrt oder Endschalter zu verwenden, aber die zusätzliche Sicherheit der Softlimits wird dadurch zunichte gemacht.

4.6.2. Voraussetzung

Die Durchführung der Referenzfahrt (engl. homing) beruht auf einigen grundlegenden Annahmen zur Maschine.

- Die negativen und positiven Richtungen basieren auf Tool Movement, die sich von der tatsächlichen Maschinenbewegung unterscheiden können. Z.B. bewegt sich bei einer Fräsmaschine typischerweise der Tisch und nicht das Werkzeug.
- Alles wird vom Nullpunkt der G53-Maschine aus referenziert, der Ursprung kann überall liegen (auch außerhalb, wo man sich bewegen kann)
- Der Nullpunkt der G53-Maschine liegt in der Regel innerhalb des Bereichs der weichen Grenzen, aber nicht zwingend.
- Der Offset des Referenzschalters legt fest, wo sich der Ursprung befindet, aber auch er wird vom Ursprung aus referenziert.
- Bei der Referenzfahrt mit Encoder-Index wird der Offset des Referenzschalters aus der Encoder-Referenzposition berechnet, nachdem der Referenzschalter ausgelöst wurde.
- Die negativen Soft(ware)-Grenzen sind das Maximum, das Sie nach der Referenzfahrt in negativer Richtung bewegen können. (aber sie sind nicht unbedingt negativ im absoluten Sinne)
- Die positiven Soft(ware)-Grenzen sind die maximale Bewegung, die Sie nach der Referenzfahrt in positiver Richtung ausführen können. (Sie sind jedoch nicht unbedingt positiv im absoluten Sinne, obwohl es üblich ist, sie als positive Zahl festzulegen)
- Soft(ware)-Grenzwerte befinden sich innerhalb des Endschalterbereichs.
- (Endgültige) Referenzpunktposition liegt innerhalb des weichen (engl. soft) Grenzbereichs
- (Bei Verwendung einer schalterbasierten Referenzfahrt nutzen die Referenzschalter entweder die Endschalter (gemeinsame Referenzfahrt-/Endschalter) oder befinden sich bei Verwendung eines separaten Referenzschalters im Bereich der Endschalter.

- Bei Verwendung eines separaten Referenzschalters ist es möglich, die Referenzfahrt auf der falschen Seite des Referenzschalters zu starten, was in Verbindung mit der Option HOME_IGNORE_LIMITS zu einem harten Absturz führen kann. Sie können dies vermeiden, indem Sie den Home-Schalter so einstellen, dass er seinen Zustand umschaltet, wenn sich die auslösende Antriebsklaue auf einer bestimmten Seite befindet, bis sie den Auslöse-Punkt wieder passiert hat. Anders ausgedrückt: Der Zustand des Home-Schalters muss die Position der Antriebsklaue relativ zum Schalter repräsentieren (d.h. *vor* oder *nach* dem Schalter), und er muss so bleiben, auch wenn die Klaue in der gleichen Richtung am Schalter vorbeiläuft.

NOTE Es ist zwar möglich, LinuxCNC mit dem G53-Maschinenursprung außerhalb der weichen Maschinengrenzen zu verwenden, aber wenn Sie G28 oder G30 verwenden, ohne die Parameter einzustellen, geht es standardmäßig zum Ursprung. Dadurch würden die Endschalter ausgelöst, bevor die Position erreicht wird.

4.6.3. Separater Home-Schalter Beispiel-Layout

Dieses Beispiel zeigt minimale und maximale Endschalter mit einem separaten Home-Schalter.

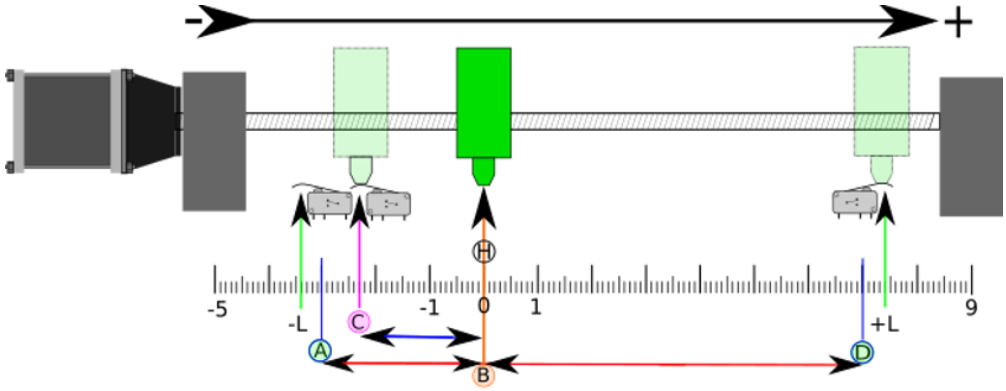


Figure 59. Demonstratives separates Schalterlayout

- A ist die negative weiche Grenze
- B ist die Koordinate der G53-Maschinen-Ursprung
- C ist der Auslösepunkt des Referenzschalters
- D ist die positive weiche Grenze
- H ist die finale Ausgangsposition (HOME) = 0 Einheiten
- Die -L und +L sind die Auslösepunkte der Endschalter
- A<→B ist die negative weiche Grenze (MIN_LIMITS) = -3 Einheiten
- B<→C ist der Home_Offset (HOME_OFFSET) = -2,3 Einheiten
- B<→D ist die positive weiche Grenze (MAX_LIMITS) = 7 Einheiten
- A<→D ist der gesamte Weg = 10 Einheiten
- Der Abstand zwischen den Endschaltern und Soft Limits (-L<→A und D←+L) wird in diesem Beispiel vergrößert
- Beachten Sie, dass zwischen den Endschaltern und dem tatsächlichen harten Kontakt für den

Auslauf nach der Deaktivierung des Verstärkers ein Abstand besteht.

NOTE Die Referenzfahrt legt das G53-Koordinatensystem fest. Der Maschinenursprung (Nullpunkt) kann an einer beliebigen Stelle liegen, aber wenn Sie den Nullpunkt auf die negative weiche Grenze setzen, werden alle G53-Koordinaten positiv, was wahrscheinlich am einfachsten zu merken ist. Dazu setzen Sie MIN_LIMIT = 0 und stellen sicher, dass MAX_LIMIT positiv ist.

4.6.4. Gemeinsamer End-/Hauptschalter Beispiel-Layout

Dieses Beispiel zeigt einen maximalen Endschalter und einen kombinierten minimalen End-/Referenzschalter.

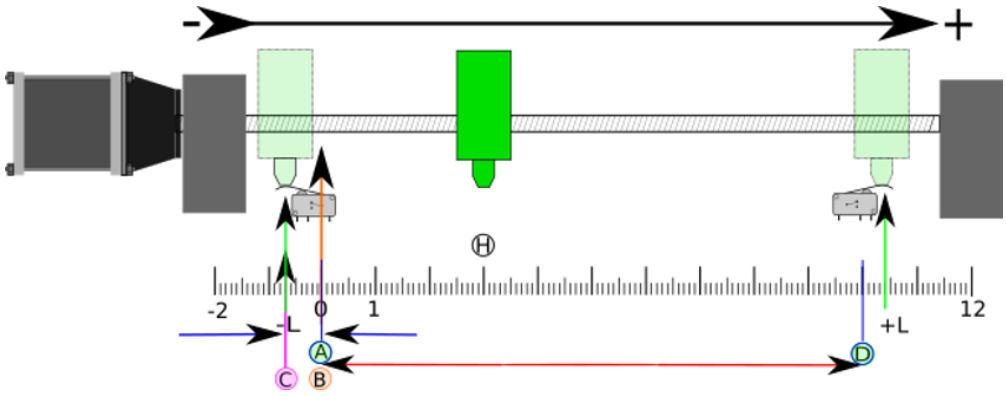


Figure 60. Beispiel für ein Layout mit geteilten Schaltern

- A ist die negative weiche Grenze.
- B ist die Koordinate der G53-Maschinen-Ursprung.
- C ist der Auslösepunkt des Referenzschalters, der gemeinsam mit dem (-L) minimalen Grenzwertauslöser verwendet wird.
- D ist die positive weiche Grenze.
- H ist die endgültige Ausgangsposition (HOME) = 3 Einheiten.
- Die -L und +L sind die Auslösepunkte der Endschalter.
- A<→B ist die negative weiche Grenze (MIN_LIMITS) = 0 Einheiten.
- B<→C ist der Home_Offset (HOME_OFFSET) = -0,7 Einheiten.
- B<→D ist die positive weiche Grenze (MAX_LIMITS) 10 Einheiten.
- A<→D ist der gesamte Weg = 10 Einheiten.
- Der Abstand zwischen den Endschaltern und den Soft Limits (-L<→A und D<→+L) wird in diesem Beispiel vergrößert.
- Beachten Sie, dass zwischen den Endschaltern und dem tatsächlichen harten Kontakt für den Auslauf nach der Deaktivierung des Verstärkers ein Abstand besteht.

4.6.5. Referenzfahrt Abfolge

Es gibt vier mögliche Referenzfahrt-Abfolgen, die durch das Vorzeichen von HOME_SEARCH_VEL und HOME_LATCH_VEL sowie die zugehörigen Konfigurationsparameter definiert sind, wie in der folgenden Tabelle dargestellt. Es gibt zwei wesentliche Varianten: HOME_SEARCH_VEL und HOME_LATCH_VEL haben das gleiche Vorzeichen oder sie haben entgegengesetzte Vorzeichen. Eine genauere Beschreibung der Funktionen der einzelnen Konfigurationsparameter finden Sie im folgenden Abschnitt.

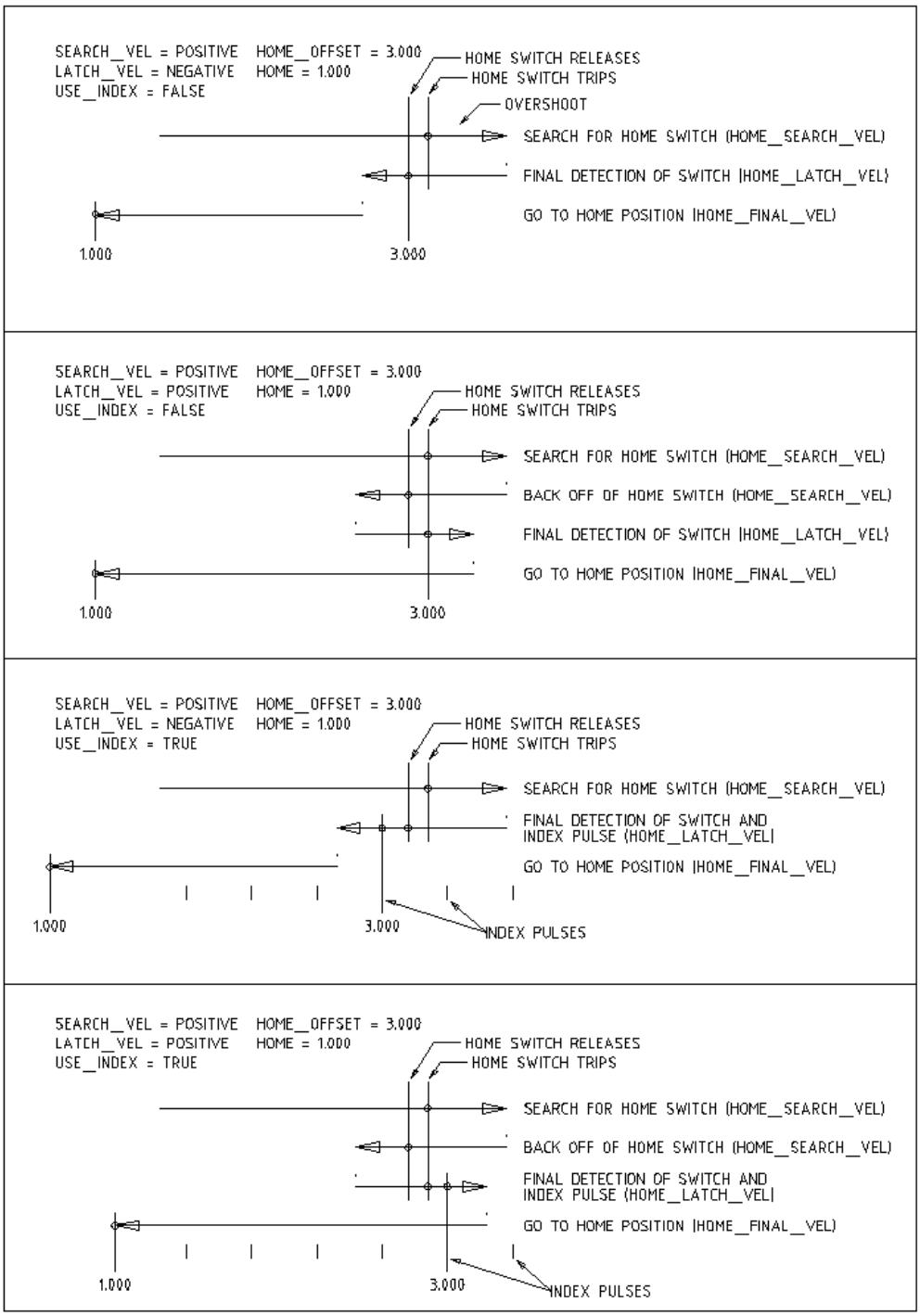


Figure 61. Referenzfahrt-Abläufe

4.6.6. Konfiguration

Im Folgenden wird genau festgelegt, wie sich die Stammfolge verhält. Sie werden in einem [JOINT_n]-

Abschnitt der INI-Datei definiert.

Referenzfahrt Typ	HOME_SEARCH_VEL	HOME_LATCH_VEL	HOME_USE_INDEX
Unmittelbar	0	0	NO
Nur-Index	0	ungleich Null	YES
Nur Schalter	ungleich Null	ungleich Null	NO
Schalter und Index	ungleich Null	ungleich Null	YES

NOTE Alle anderen Kombinationen können zu einem Fehler führen.

HOME_SEARCH_VEL

Diese Variable hat die Einheit von Maschineneinheiten pro Sekunde.

Der Standardwert ist Null. Ein Wert von Null bewirkt, dass LinuxCNC davon ausgeht, dass es keine Home-Schalter gibt; die Suche Phase der Referenzfahrt wird übersprungen.

Wenn HOME_SEARCH_VEL ungleich Null ist, dann nimmt LinuxCNC an, dass es einen Referenzschalter (engl. home switch) gibt. Es beginnt mit der Überprüfung, ob der Referenzschalter bereits ausgelöst hat. Wenn dies der Fall ist, wird der Schalter bei HOME_SEARCH_VEL zurückgesetzt. Die Richtung des Zurückfahrens ist entgegengesetzt dem Vorzeichen von HOME_SEARCH_VEL. Anschließend wird der Schalter in der durch das Vorzeichen von HOME_SEARCH_VEL festgelegten Richtung mit einer durch den Absolutwert bestimmten Geschwindigkeit gesucht. Wenn der Referenzschalter erkannt wird, hält das Gelenk so schnell wie möglich an, wobei jedoch immer ein gewisses Überschwingen auftritt. Das Ausmaß des Überschwingens hängt von der Geschwindigkeit ab. Ist sie zu hoch, kann das Gelenk so weit überschwingen, dass es gegen einen Endschalter stößt oder gegen das Ende des Verfahrwegs prallt. Ist HOME_SEARCH_VEL hingegen zu niedrig, kann die Referenzfahrt sehr lange dauern.

HOME_LATCH_VEL

Diese Variable hat die Einheit von Maschineneinheiten pro Sekunde.

Legt die Geschwindigkeit und Richtung, die LinuxCNC verwendet, wenn es seine endgültige genaue Bestimmung der Home-Schalter (falls vorhanden) und Index-Impuls Lage (falls vorhanden) macht. Es wird in der Regel langsamer als die Suchgeschwindigkeit sein, um die Genauigkeit zu maximieren. Wenn HOME_SEARCH_VEL und HOME_LATCH_VEL das gleiche Vorzeichen haben, dann wird die Latch-Phase durchgeführt, während man sich in die gleiche Richtung wie die Suchphase bewegt. (In diesem Fall fährt LinuxCNC zunächst vom Schalter zurück, bevor es sich mit der Verriegelungsgeschwindigkeit wieder auf ihn zubewegt). Wenn HOME_SEARCH_VEL und HOME_LATCH_VEL entgegengesetzte Vorzeichen haben, wird die Latch-Phase durchgeführt, während man sich in die entgegengesetzte Richtung der Suchphase bewegt. Das bedeutet, dass LinuxCNC den ersten Impuls einrastet, nachdem es den Schalter verlassen hat. Wenn HOME_SEARCH_VEL gleich Null ist (d.h. es gibt keinen Home-Schalter), und dieser Parameter ungleich Null ist, geht LinuxCNC zur Index-Impuls-Suche über. Wenn HOME_SEARCH_VEL nicht Null ist und dieser Parameter ist auf Null gesetzt, so ist es ein Fehler und die

Referenzfahrt wird entsprechend fehlschlagen. Der Standardwert ist Null.

HOME_FINAL_VEL

Diese Variable hat die Einheit von Maschineneinheiten pro Sekunde.

Sie gibt die Geschwindigkeit an, die LinuxCNC verwendet, wenn es seine Bewegung von HOME_OFFSET zur HOME-Position durchführt. Wenn die HOME_FINAL_VEL in der INI-Datei fehlt, dann wird die maximale Gelenkgeschwindigkeit verwendet, um diese Bewegung zu machen. Der Wert muss eine positive Zahl sein.

HOME_SEARCH_DIST

This variable has units of machine-units.

The farthest the search move may travel without finding the home switch. When the switch is not seen within this distance the joint stops there and the homing is aborted with an error; the next attempt searches on from that point. The default value is zero: the search move is bounded only by twice the joint's travel, so a dead switch takes the joint to its limit switch, or into the end of its travel when the limit switch is shared with the home switch and ignored. On a continuous rotary joint, whose soft limits are set far apart, this is the only thing that stops a search with a dead switch: set it to a little over one revolution.

HOME_LATCH_DIST

This variable has units of machine-units.

The farthest any of the moves made next to the home switch may travel: the back-off move that waits for the switch to clear, the latch move that finds the switch again, and the index pulse search. When the move ends without result the joint moves back to where that move began, at the speed of that move, and the homing is aborted with an error. The default value is zero: these moves are bounded only by twice the joint's travel. A switch stuck on sends the back-off move away from the switch for the whole travel; a switch that fails after its first trip sends the latch move the same way at latch speed. A value of a few times the distance the joint overshoots when the switch trips (HOME_SEARCH_VEL squared over twice the joint acceleration) plus the width of the switch's active zone bounds both. With HOME_USE_INDEX the value must also cover one revolution of the encoder.

When a joint of a synchronized sequence (negative HOME_SEQUENCE) runs out of this distance, every joint of that sequence stops and each one moves back to where its own current move began, so the joints end where they were when the move started.

HOME_IGNORE_LIMITS

Kann die Werte YES / NO annehmen. Der Standardwert für diesen Parameter ist NO. Dieses Flag bestimmt, ob LinuxCNC die Endschaltereingabe für dieses Gelenk während der Referenzfahrt ignoriert. Diese Einstellung wird nicht ignorieren Endschalter Eingänge für andere Gelenke. Wenn Sie keinen separaten Refrenzschalter haben, setzen Sie diesen Parameter auf YES und verbinden Sie das Endschaltersignal mit dem gemeinsamen Referenz-(engl. Home-)schalter-Eingang in HAL. LinuxCNC

wird den Endschaltereingang für dieses Gelenk während der Referenzfahrt ignorieren. Um nur einen Eingang für alle Referenzfahrten und Endschalter zu verwenden, müssen Sie die Endschaltersignale der Gelenke, die nicht in HAL referenzieren, blockieren und ein Gelenk nach dem anderen referenzieren.

HOME_USE_INDEX

Gibt an, ob es einen Indeximpuls gibt oder nicht. Wenn das Flag wahr ist (HOME_USE_INDEX = YES), wird LinuxCNC auf die steigende Flanke des Index-Impulses einrasten. Wenn falsch, wird LinuxCNC entweder auf die steigende oder die fallende Flanke des Home-Schalters einrasten (abhängig von den Vorzeichen von HOME_SEARCH_VEL und HOME_LATCH_VEL). Der Standardwert ist NO.

NOTE HOME_USE_INDEX erfordert Verbindungen in Ihrer HAL-Datei zu `joint.n.index-enable` vom `encoder.n.index-enable`.

HOME_INDEX_NO_ENCODER_RESET

Voreinstellung ist NO. Verwenden Sie YES, wenn der für dieses Gelenk verwendete Encoder seinen Zähler nicht zurücksetzt, wenn ein Indeximpuls nach der Aktivierung des Gelenkindex_enable HAL-Pins erkannt wird. Gilt nur für HOME_USE_INDEX = YES.

HOME_OFFSET

Definiert die Lage des Ursprungsnullpunkts des G53-Maschinenkoordinatensystems. Es ist der Abstand (Offset), in gemeinsamen Einheiten, von der Maschine Ursprung auf die Referenzsschalter Auslösepunkt oder Index-Impuls. Nach der Erkennung der Schalter Auslösepunkt / Index-Impuls, setzt LinuxCNC die gemeinsame Koordinatenposition zu HOME_OFFSET, und damit die Definition der Ursprungs, von dem sich die weichen Grenzen ableiten. Der Standardwert ist Null.

NOTE Die Position des Referenzschalters, die durch die Variable HOME_OFFSET angegeben wird, kann innerhalb oder außerhalb der Soft Limits liegen. Sie werden gemeinsam mit oder innerhalb der harten Endschalter verwendet.

HOME

Die Position, die das Gelenk nach Abschluss der Referenzierungsfahrt annehmen soll. Nach der Erkennung der Referenzschalter oder erst des Referenzschalters gefolgt vom Index-Impuls (je nach Konfiguration), und die Einstellung der Koordinate dieses Punktes zu HOME_OFFSET, führt LinuxCNC zu HOME als Abschluss Referenzfahrt durch. Der Standardwert ist Null. Beachten Sie, dass, selbst wenn dieser Parameter der gleiche wie HOME_OFFSET ist, das Gelenk vermutlich leicht über die verriegelte Position hinaus bewegt worden. Daher wird es zu diesem Zeitpunkt immer eine kleine Bewegung geben (es sei denn, HOME_SEARCH_VEL ist Null und die gesamte Such-/Speicherphase wurde übersprungen). Diese letzte Bewegung wird mit der maximalen Geschwindigkeit des Gelenks ausgeführt, es sei denn, HOME_FINAL_VEL wurde eingestellt.

NOTE Der Unterschied zwischen *HOME_OFFSET* und *HOME* besteht darin, dass *HOME_OFFSET* zunächst die Ursprungsposition und den Maßstab auf der Maschine festlegt, indem der

HOME_OFFSET -Wert auf die Position angewendet wird, an der die Ausgangsposition gefunden wurde, und dann *HOME* angibt, wohin sich das Gelenk auf diesem Maßstab bewegen soll.

HOME_IS_SHARED

Wenn es keinen separaten Referenzschaltereingang für diese Gelenk gibt, sondern mehrere Taster an denselben Pin angeschlossen sind, setzen Sie diesen Wert auf 1, um zu verhindern, dass die Referenzfahrt beginnt, wenn einer der gemeinsamen Schalter bereits geschlossen ist. Setzen Sie diesen Wert auf 0, um die Referenzfahrt zu ermöglichen, auch wenn der Schalter bereits geschlossen ist.

HOME_ABSOLUTE_ENCODER

Verwendung für absolute Encoder. In Reaktion auf eine Anforderung zur Referenzfahrt des Gelenks wird die aktuelle Gelenkposition auf den *[JOINT_n]HOME_OFFSET* Wert gesetzt.

Die abschließende Bewegung zur *[JOINT_n]HOME* Position ist entsprechend der *HOME_ABSOLUTE_ENCODER* Einstellung optional:

```
HOME_ABSOLUTE_ENCODER = 0 (Standard) Gelenk verwendet keinen Absolutwertgeber
HOME_ABSOLUTE_ENCODER = 1 Absolutwertgeber, endgültige Bewegung zu [JOINT_n]HOME
HOME_ABSOLUTE_ENCODER = 2 Absolutwertgeber, KEINE endgültige Bewegung zu [JOINT_n]HOME
```

NOTE Eine HOME_IS_SHARED-Einstellung wird stillschweigend ignoriert.

NOTE Eine Aufforderung, für ein Gelenk die Referenzfahrt zu wiederholen, wird stillschweigend ignoriert.

HOME_SEQUENCE

Wird verwendet, um eine Multigelenk-Referenzierungssequenz **HOME ALL** zu definieren und die Referenzierungsreihenfolge zu erzwingen (z.B. darf Z nicht referenziert werden, wenn X noch nicht referenziert ist). Ein Gelenk kann erst dann referenziert werden, wenn alle Gelenke mit einer niedrigeren (absoluten) HOME_SEQUENCE bereits referenziert wurden und sich am HOME_OFFSET befinden. Wenn zwei Gelenke die gleiche HOME_SEQUENCE haben, können sie gleichzeitig referenziert werden.

NOTE Wenn HOME_SEQUENCE nicht angegeben ist, wird das Gelenk nicht durch die **HOME ALL**-Sequenz referenziert (sondern kann durch einzelne gelenkspezifische Referenzierungsbefehle referenziert werden).

Die anfängliche HOME_SEQUENCE-Nummer kann 0, 1 (oder -1) sein. Der absolute Wert der Sequenznummern muss um eins erhöht werden - das Überspringen von Sequenznummern wird nicht unterstützt. Wenn eine Sequenznummer weggelassen wird, stoppt **HOME ALL** die Referenzfahrt nach Abschluss der letzten gültigen Sequenznummer.

Negative HOME_SEQUENCE-Werte zeigen an, dass die Gelenke in der Sequenz die letzte Bewegung zu

[JOINT_n]HOME **synchronisieren** sollen, indem sie warten, bis alle Gelenke in der Sequenz hierzu bereit sind. Wenn ein Gelenk einen **negativen** HOME_SEQUENCE-Wert hat, dann müssen alle Gelenke mit demselben absoluten Wert (positiv oder negativ) des HOME_SEQUENCE-Wertes die letzte Bewegung synchronisieren.

Eine **negative** HOME_SEQUENCE gilt auch für das Ausführen einer Referenzfahrt eines einzelnen Gelenks. Wenn der HOME_SEQUENCE-Wert **negativ** ist, werden alle Gelenke, die den gleichen absoluten Wert dieser HOME_SEQUENCE haben, **gemeinsam mit einer synchronisierten Endbewegung** freigesetzt. Wenn der HOME_SEQUENCE-Wert Null oder positiv ist, wird nur das angegebene Gelenk in die Ausgangsstellung gebracht.

Das manualle Bewegen im "joint mode" von Gelenken mit einer negativen HOME_SEQUENCE ist nicht zulässig. Bei üblichen Portalanwendungen kann ein solches Verfahren zu einer Fehlausrichtung führen (Racking). Beachten Sie, dass das konventionelle Jogging in Weltkoordinaten immer verfügbar ist, sobald eine Maschine referenziert ist.

Beispiele für ein 3-Gelenk-System

Zwei Sequenzen (0,1), keine Synchronisation

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = 1
[JOINT_2]HOME_SEQUENCE = 1
```

Zwei Sequenzen, Gelenke 1 und 2 synchronisiert

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

Bei gemischten positiven und negativen Werten synchronisierten die Gelenke 1 und 2

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE =  1
```

Eine Sequenz, keine Synchronisation

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE =  0
[JOINT_2]HOME_SEQUENCE =  0
```

Eine Sequenz, alle Gelenke synchronisiert

```
[JOINT_0]HOME_SEQUENCE = -1
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

VOLATILE_HOME

Wenn diese Einstellung aus TRUE gesetzt ist, geht für dieses Gelenk die Referenzeinstellung nicht verloren, wenn die Maschine in den AUS-Zustand übergeht. Dies ist für jedes Gelenk geeignet, das seine Position nicht beibehält, wenn der Gelenkantrieb ausgeschaltet ist. Einige Schrittantriebe, insbesondere Mikroschrittantriebe, können dies benötigen.

LOCKING_INDEXER

Handelt es sich bei diesem Gelenk um einen verriegelnden Drehindexer, wird es vor der Referenzfahrt entriegelt und danach verriegelt.

Unmittelbare Referenzfahrt

Wenn ein Gelenk keine Home-Schalter oder keine logische Home-Position wie ein Drehgelenk hat und Sie möchten, dass dieses Gelenk an der aktuellen Position startet, wenn die Schaltfläche "Home All" in der AXIS GUI gedrückt wird, dann sind die folgenden INI-Einträge für dieses Gelenk erforderlich.

```
HOME_SEARCH_VEL = 0
HOME_LATCH_VEL = 0
HOME_USE_INDEX = NO
HOME_OFFSET = 0 (oder der Offset der Ausgangsposition (HOME))
HOME_SEQUENCE = 0 (oder eine andere gültige Sequenznummer)
```

NOTE

Die Standardwerte für nicht spezifizierte HOME_SEARCH_VEL, HOME_LATCH_VEL, HOME_USE_INDEX, HOME und HOME_OFFSET sind **Null**, so dass sie weggelassen werden können, wenn eine sofortige Referenzfahrt angefordert wird. Eine gültige HOME_SEQUENCE-Nummer sollte in der Regel angegeben werden, da das Weglassen einer HOME_SEQUENCE die Verbindung vom **HOME ALL**-Verhalten ausschließt (siehe oben).

Referenzfahrt verhindern

Ein HAL -Pin (motion.homing-inhibit) ist vorgesehen, um die Einleitung der Referenzfahrt sowohl für "alle Achsen gleichzeitig (engl. "Home All") als auch für die Referenzfahrt einzelner Gelenke zu unterbinden.

Einige Systeme nutzen die Bestimmungen für die Synchronisierung der endgültigen Gelenkbewegungen, die werden durch negative [JOINT_N]HOME_SEQUENCE=INI-Dateielemente. Standardmäßig verbieten die Synchronisierungsbestimmungen ein **Gelenk**-Jogging vor der Referenzfahrt, um ein **Gelenk**-Jogging zu verhindern, das die Maschine falsch ausrichten könnte (z. B. Portalkreuzung).

Der Systemintegrator kann das **Gelenk**-Jogging vor der Referenzfahrt mit einer HAL-Logik erlauben, um die [JOINT_N]HOME_SEQUENCE-Elemente umzuschalten. Diese Logik sollte auch den Pin **motion.homing-inhibit** aktivieren, um sicherzustellen, dass die Referenzfahrt nicht versehentlich eingeleitet wird, wenn der **Joint**-Jogging-Modus aktiviert ist.

Beispiel: Synchronisierte Gelenke 0,1 mit negativer Sequenz (-1) für synchronisierte Referenzfahrt mit

einem Schalter (allow_jjog), der eine positive Sequenz (1) für individuelles **Gelenk**-Jogging vor der Referenzfahrt wählt (partieller HAL-Code):

```
loadrt mux2           names=home_sequence_mux
loadrt conv_real_sint names=home_sequence_sint
setp home_sequence_mux.in0 -1
setp home_sequence_mux.in1  1
addf home_sequence_mux servo-thread
addf home_sequence_sint servo-thread
...
net home_seq_float <= home_sequence_mux.out
net home_seq_float => home_sequence_sint.in
net home_seq_sint  <= home_sequence_sint.out
net home_seq_sint  => ini.0.home_sequence
net home_seq_sint  => ini.1.home_sequence
...
# allow_jjog: pin created by a virtual panel or hardware switch
net hsequence_select <= allow_jjog
net hsequence_select => home_sequence_mux.sel
net hsequence_select => motion.homing-inhibit
```

NOTE INI HAL-Pins (wie ini.N.home_sequence) sind nicht verfügbar, bis milltask startet, so dass die Ausführung der oben genannten HAL-Befehle mit Hilfe einer postgui HAL-Datei oder eines verzögerten [APPLICATION]APP=-Skripts verschoben werden sollte.

NOTE Für die Echtzeitsynchronisation des Gelenk-Joggings für mehrere Gelenke sind zusätzliche HAL-Verbindungen für die Jog-Pins vom Typ Manual-Pulse-Generator (MPG) erforderlich (`joint.N.enable`, `joint.N.scale`, `joint.N.counts`).

Eine Beispielsimulationskonfiguration (gantry_jjog.ini), die das Joggen der Gelenke bei Verwendung negativer Nullpunktsequenzen demonstriert, befindet sich im Verzeichnis: configs/sim/axis/gantry/.

4.7. Konfiguration der Drehmaschine

4.7.1. Standard-Ebene

Als der Interpreter für LinuxCNC geschrieben wurde, war dieser für Fräsmaschinen konzipiert. Deshalb ist die Standard-Ebene XY (G17). Eine normale Drehmaschine verwendet jedoch die XZ-Ebene (G18). Um die Standardebene zu ändern, fügen Sie die folgende Zeile in die INI-Datei im Abschnitt RS274NGC ein.

```
RS274NGC_STARTUP_CODE = G18
```

Die obigen Angaben können in einem G-Code-Programm überschrieben werden, daher sollten Sie wichtige Dinge immer in der Präambel der G-Code-Datei festlegen.

4.7.2. INI-Einstellungen

Die folgenden INI-Einstellungen werden für den Drehmaschinenmodus in Axis zusätzlich zu den

normalen Einstellungen in der INI-Datei benötigt oder ersetzen diese. Diese historischen Einstellungen verwenden die Identitätskinematik (trivkins) und *drei* Gelenke (0,1,2) entsprechend den Koordinaten x, y, z. Das Gelenk 1 für die unbenutzte y-Achse ist erforderlich, wird aber in diesen historischen Konfigurationen nicht verwendet. Simulierte Drehmaschinen-Konfigurationen können diese historischen Einstellungen verwenden. GMOCCAPY verwendet ebenfalls die erwähnten Einstellungen, bietet aber zusätzliche Einstellungen, siehe den Abschnitt GMOCCAPY für Details.

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
# ...

[KINS]
KINEMATICS = trivkins
JOINTS = 3

[TRAJ]
COORDINATES = X Z
# ...

[JOINT_0]
# ...
[JOINT_2]
# ...
[AXIS_X]
# ...
[AXIS_Z]
# ...
```

Mit der Einbindung von joints_axes kann eine einfachere Konfiguration mit nur den beiden benötigten Gelenken vorgenommen werden, indem trivkins mit dem Parameter *coordinates*= angegeben wird:

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
# ...

[KINS]
KINEMATICS = trivkins coordinates=xz
JOINTS = 2

[TRAJ]
COORDINATES = X Z
# ...

[JOINT_0]
# ...
[JOINT_1]
# ...
[AXIS_X]
# ...
[AXIS_Z]
# ...
```

4.8. Stepper Schnellstart

Dieser Abschnitt geht davon aus, dass Sie eine Standardinstallation von der Live-CD durchgeführt haben. Nach der Installation wird empfohlen, den Computer mit dem Internet zu verbinden und darauf zu warten, dass der Update-Manager erscheint, um die neuesten Updates für LinuxCNC und Ubuntu zu erhalten, bevor Sie fortfahren.

4.8.1. Latenz-Test

Der Latenztest bestimmt, wie spät Ihr Computerprozessor auf eine Anfrage reagiert. Manche Hardware kann die Verarbeitung unterbrechen, was beim Betrieb einer CNC-Maschine zu verpassten Schritten führen kann. Dies ist der erste Schritt, den Sie tun müssen. Folgen Sie den Anweisungen hier, um den Latenztest durchzuführen.

4.8.2. Sherline

Wenn Sie eine Sherline haben, sind mehrere vordefinierte Konfigurationen vorhanden. Diese finden Sie im Hauptmenü CNC/EMC. Wählen Sie dann die Sherline-Konfiguration, die Ihrer entspricht, und speichern Sie eine Kopie.

4.8.3. Xylotex

Wenn Sie eine Xylotex haben, können Sie die folgenden Abschnitte überspringen und gehen Sie direkt auf die Schrittmotor Konfigurations-Assistenz (engl. Stepper Config Wizard). LinuxCNC hat schnelle Einrichtung für die Xylotex Maschinen zur Verfügung gestellt.

4.8.4. Maschineninformationen

Sammeln Sie die Informationen über jede Achse Ihrer Maschine.

Das Timing des Antriebs ist in Nanosekunden angegeben. Wenn Sie sich über das Timing unsicher sind, so sind viele gängige Antriebe bereits durch den Stepper-Konfigurationsassistenten beschrieben. Beachten Sie einige neuere Gecko-Antriebe ein anderes Timing haben als das Original. Ein Liste ist auch auf der Benutzer gepflegt LinuxCNC Wiki-Site von mehr Laufwerke.

Achse	Treiber-Typ	Schrittzeit (ns)	Schwittweite (ns)	Dir. Hold (ns)	Dir. Setup (ns)
X					
Y					
Z					

4.8.5. Informationen zur Pinbelegung

Sammeln Sie die Informationen über die Verbindungen zwischen Ihrem Rechner und dem parallelen

PC-Anschluss.

Ausgangs-Pin	Typ. Funktion	Wenn Unterschiedlich	Input Pin	Typ. Funktion	Wenn Unterschiedlich
1	E-Stop Out		10	X End-/Referenzschalter	
2	X Schritt		11	Y End-/Referenzschalter	
3	X Richtung		12	Z End-/Referenzschalter	
4	Y-Schritt		13	A End-/Referenzschalter	
5	Y-Richtung		15	Sonde In	
6	Z Schritt				
7	Z Richtung				
8	A Schritt				
9	A Richtung				
14	Spindel Uhrzeigersinn				
16	Spindel PWM				
17	Verstärker Aktivieren				

Beachten Sie, dass alle nicht verwendeten Pins in der Dropdown-Box auf Unused gesetzt werden sollten. Diese können später jederzeit geändert werden, indem StepConf erneut ausgeführt wird.

4.8.6. Mechanische Informationen

Sammeln Sie Informationen über Schritte und Getriebe. Das Ergebnis sind Schritte pro Benutzereinheit, die für SCALE in der INI-Datei verwendet werden.

Achse	Schritte/Umdr.	Mikro-Schritte	Motor Verzahnung	Leitspindel Zähne	Steigung der Leitspindel
X					

Achse	Schritte/Umdr.	Mikro-Schritte	Motor Verzahnung	Leitspindel Zähne	Steigung der Leitspindel
Y					
Z					

- *Schritte pro Umdrehung* - gibt an, wie viele Schritte der Schrittmotor für eine Umdrehung benötigt. Typisch sind 200.
- Micro Steps' - gibt an, wie viele Schritte der Antrieb benötigt, um den Schrittmotor einen vollen Schritt zu bewegen. Wenn kein Microstepping verwendet wird, ist diese Zahl 1. Wenn Microstepping verwendet wird, hängt der Wert von der Hardware des Schrittmotors ab.
- *Motor Teeth and Leadscrew Teeth* - ist, wenn Sie eine Untersetzung (Zahnrad, Kette, Zahnriemen usw.) zwischen Motor und Leitspindel haben. Wenn nicht, setzen Sie beide auf 1.
- Leitspindelsteigung' - gibt an, wie viel Bewegung (in Benutzereinheiten) in einer Leitspindelumdrehung stattfindet. Wenn Sie auf Zoll eingestellt sind, ist es Zoll pro Umdrehung. Wenn Sie in Millimetern einstellen, sind es Millimeter pro Umdrehung.

Das Nettoergebnis, nach dem Sie suchen, ist die Anzahl der CNC-Ausgabeschritte, die erforderlich sind, um eine Benutzereinheit (Zoll oder mm) zu bewegen.

Example 1. Einheiten Zoll

```
Stepper = 200 Schritte pro Umdrehung
Antrieb = 10 Mikroschritte pro Schritt
Motorverzahnung = 20
Leitspindelzähne = 40
Steigung der Leitspindel = 0,2000 Zoll pro Umdrehung
```

Aus den obigen Angaben geht hervor, dass sich die Leitspindel um 0,200 Zoll pro Umdrehung bewegt. - Der Motor dreht sich 2.000 Mal pro 1 Spindeldrehung. - Der Antrieb benötigt 10 Mikroschritt-Eingänge, um den Schrittmotor einen vollen Schritt zu bewegen. - Der Antrieb benötigt 2000 Schritte für den Stepper für eine vollständige Umdrehung.

Die erforderliche Skala lautet also:

$$\frac{200 motor steps}{1 motor rev} \times \frac{10 microsteps}{1 motor step} \times \frac{2 motor revs}{1 leadscrew rev} \times \frac{1 leadscrew revs}{0.2000 inch} = \frac{20{,}000 microsteps}{inch}$$

Example 2. Einheiten mm

```
    Stepper = 200 Schritte pro Umdrehung
    Antrieb = 8 Mikroschritte pro Schritt
    Motorverzahnung = 30
    Leitspindelzähne = 90
    Gewindespindelsteigung = 5,00 mm pro Umdrehung
```

Aus den oben genannten Informationen: - Die Leitspindel bewegt sich 5,00 mm pro Umdrehung. - Der Motor dreht sich 3.000 Mal pro 1 Umdrehung der Leitspindel. - Der Antrieb benötigt 8 Mikroschritt-Eingänge, um den Schrittmotor einmal zu bewegen. - Der Antrieb benötigt 1600 Schritte, um den Stepper eine Umdrehung zu drehen.

Die erforderliche Skala lautet also:

$$\frac{200\,full\,steps}{1\,rev} x \frac{8\,microsteps}{1\,step} x \frac{3\,revs}{1\,leadscrew\,rev} x \frac{1\,leadscrew\,rev}{5.00\text{mm}} = \frac{960\,steps}{1\,mm}$$

4.9. Schrittmotor-Konfiguration

4.9.1. Einführung

Die bevorzugte Methode zum Einrichten einer Standard-Maschine mit Schrittmotoren (engl. stepper machine) ist der Stepper-Konfigurations-Assistent (engl. stepper configuration wizard). Siehe das Kapitel Stepper-Konfigurations-Assistenz.

In diesem Kapitel werden einige der gängigsten Einstellungen für die manuelle Einrichtung eines schrittmotorbasierten Systems beschrieben. Diese Systeme verwenden Schrittmotoren mit Antrieben, die Schritt- und Richtungssignale akzeptieren.

Es ist eines der einfacheren Systeme, da die Motoren im offenen Regelkreis laufen (keine Rückmeldung von den Motoren), aber das System muss richtig konfiguriert werden, damit die Motoren nicht abgewürgt werden oder Schritte verlieren.

Der größte Teil dieses Kapitels basiert auf einer Beispielkonfiguration, die zusammen mit LinuxCNC veröffentlicht wurde. Die Konfiguration heißt stepper_inch, und kann durch Ausführen der Konfigurations-Auswahl (engl. configuration picker) gefunden werden.

4.9.2. Maximale Schrittgeschwindigkeit

Bei der Software-Schrittgenerierung beträgt die maximale Schrittrate einen Schritt pro zwei BASE_PERIODs für die Schritt- und Richtungsausgabe. Die maximal geforderte Schrittgeschwindigkeit ist das Produkt aus MAX_VELOCITY und INPUT_SCALE einer Achse. Wenn die geforderte Schrittgeschwindigkeit nicht erreicht werden kann, kommt es zu folgenden Fehlern, insbesondere bei Eilgängen und G0-Bewegungen.

Wenn Ihr Stepper-Treiber Quadratur-Eingänge akzeptieren kann, verwenden Sie diesen Modus. Mit einem Quadratursignal ist ein Schritt pro BASE_PERIOD möglich, wodurch sich die maximale Schrittrate verdoppelt.

Andere Abhilfemaßnahmen sind die Verringerung einer oder mehrerer der folgenden Einstellungen: BASE_PERIOD (eine zu niedrige Einstellung führt dazu, dass die Maschine nicht mehr reagiert oder sogar blockiert), INPUT_SCALE (wenn Sie verschiedene Schrittgrößen auf Ihrem Stepper-Treiber auswählen können, das Verhältnis der Riemenscheiben oder die Spindelsteigung ändern) oder MAX_VELOCITY und STEPGEN_MAXVEL.

Wenn keine gültige Kombination von BASE_PERIOD, INPUT_SCALE und MAX_VELOCITY akzeptabel ist, dann sollten Sie die Hardware-Schritterzeugung in Betracht ziehen (z. B. mit den von LinuxCNC unterstützten Universal Stepper Controller, Mesa-Karten und anderen).

4.9.3. Pinbelegung

Einer der größten Mängel in EMC war, dass man die Pinbelegung nicht ohne Neukompilierung des Quellcodes angeben konnte. EMC2 war viel flexibler, und jetzt in LinuxCNC (dank der Hardware Abstraction Layer) können Sie leicht angeben, welches Signal welchen Weg nimmt. Siehe die HAL Grundlagen für weitere Informationen über HAL.

Wie in der HAL-Einführung und im Tutorial beschrieben, haben wir Signale, Pins und Parameter innerhalb des HAL.

NOTE Wir stellen nur eine Achse vor, um uns kurz zu fassen, alle anderen sind ähnlich.

Die für unsere Pinbelegung relevanten sind:

```
Signale: Xstep, Xdir & Xen
Pins: parport.0.pin-XX-out & parport.0.pin-XX-in
```

Je nachdem, was Sie in Ihrer INI-Datei ausgewählt haben, verwenden Sie entweder standard_pinout.hal oder xylotex_pinout.hal. Dies sind zwei Dateien, die den HAL anweisen, wie die verschiedenen Signale & Pins zu verbinden sind. Weiter unten werden wir uns mit der standard_pinout.hal beschäftigen.

Standard-Pinbelegung HAL

Diese Datei enthält mehrere HAL-Befehle und sieht normalerweise wie folgt aus:

```
# Standard-Pinout-Konfigurationsdatei für 3-Achsen-Stepper
# Verwendung eines Parports für E/A
#
# zuerst den Parport-Treiber laden
loadrt hal_parport cfg="0x0378"
#
# als nächstes die Parport-Funktionen mit den Threads verbinden
# lese zuerst die Eingänge
addf parport.0.read base-thread 1
# Ausgaben zuletzt schreiben
addf parport.0.write base-thread -1
#
#  schließlich physische Pins mit den Signalen verbinden Netz
net Xstep => parport.0.pin-03-out
net Xdir  => parport.0.pin-02-out
net Ystep => parport.0.pin-05-out
net Ydir  => parport.0.pin-04-out
net Zstep => parport.0.pin-07-out
net Zdir  => parport.0.pin-06-out

# Signal für den Estop-Loopback erzeugen
net estop-loop iocontrol.0.user-enable-out iocontrol.0.emc-enable-in
```

```
# Signale für die Werkzeugladeschleife erzeugen
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed

# "spindle on" Bewegungssteuerungs-Pin mit einem physischen Pin verbinden
net spindle-on spindle.0.on => parport.0.pin-09-out

###
### Sie könnten etwas wie das folgende verwenden, um Chopper-Antriebe zu aktivieren, wenn
die Maschine eingeschaltet ist
### Das Xen-Signal wird in core_stepper.hal definiert.
###

# net Xen => parport.0.pin-01-out

###
### Wenn Sie für diesen Pin einen aktiven low-Wert wünschen, invertieren Sie ihn wie
folgt:
###

# setp parport.0.pin-01-out-invert 1

###
### Ein Beispiel für einen Referenzschalter (engl. home switch) an der X-Achse (Achse 0).
Erzeugen Sie ein Signal,
### verbinden Sie den eingehenden Parport-Pin mit dem Signal, dann verbinden Sie das
Signal
### mit dem LinuxCNC's Achse 0 Referenzschalter Eingabe-Pin.
###

# net Xhome parport.0.pin-10-in => joint.0.home-sw-in

###
### Geteilte Referenzschalter alle zu einem einzelnen parallel port Pin führen?
### Das ist ok, nutzen Sie das gleiche Signal an allen Achsen, aber stellen Sie sicher,
dass Sie
### HOME_IS_SHARED und HOME_SEQUENCE in der INI-Datei. setzen.
###

# net homeswitches <= parport.0.pin-10-in
# net homeswitches => joint.0.home-sw-in
# net homeswitches => joint.1.home-sw-in
# net homeswitches => joint.2.home-sw-in

###
### Beispiel für separate Endschalter auf der X-Achse (Achse 0)
###

# net X-neg-limit parport.0.pin-11-in => joint.0.neg-lim-sw-in
# net X-pos-limit parport.0.pin-12-in => joint.0.pos-lim-sw-in

###
### Genau wie beim Beispiel der gemeinsamen Referenzschalter können Sie auch
### Endschalter miteinander verbinden.  Achten Sie darauf, wenn Sie einen auslösen, wird
LinuxCNC stoppen,
```

```
### kann Ihnen aber nicht sagen, welche Schalter/Achse verantwortlich ist.  Seien Sie
vorsichtig, wenn die den Betrieb
### von dieser Extremposition wieder aufnehmen.
###

# net Xlimits parport.0.pin-13-in => joint.0.neg-lim-sw-in joint.0.pos-lim-sw-in
```

Die Zeilen, die mit # beginnen, sind Kommentare, die lediglich dazu dienen, den Leser durch die Datei zu führen.

Übersicht

Es gibt eine Reihe von Operationen, die ausgeführt werden, wenn die Datei standard_pinout.hal ausgeführt/interpretiert wird:

- Der Parallel-Port (kurz Parport)-Treiber wird geladen (siehe das Parport Kapitel für Details).
- Die Lese- und Schreibfunktionen des Parport-Treibers werden dem Basis-Thread zugewiesen [3].
- Die Schritt & Richtungssignale für die Achsen X, Y, Z werden mit Pins auf dem Parport verbunden.
- Weitere I/O-Signale werden angeschlossen (Notaus Loopback, Werkzeugwechsler Loopback).
- Ein Spindel-Ein-Signal wird definiert und mit einem Parport-Pin verbunden.

Ändern der Datei standard_pinout.hal

Wenn Sie die Datei standard_pinout.hal ändern möchten, benötigen Sie lediglich einen Texteditor. Öffnen Sie die Datei und suchen Sie die Teile, die Sie ändern möchten.

Wenn Sie z.B. den Pin für die X-Achse Step & Directions (engl. für Schritt & Richtung) Signale ändern wollen, müssen Sie nur die Nummer im *parport.0.pin-XX-out* Namen ändern:

```
net Xstep parport.0.pin-03-out
net Xdir  parport.0.pin-02-out
```

kann geändert werden in:

```
net Xstep parport.0.pin-02-out
net Xdir  parport.0.pin-03-out
```

oder grundsätzlich jeden andere *out* Pin, die Sie mögen.

Tipp: Achten Sie darauf, dass Sie nicht mehr als ein Signal an denselben Pin anschließen.

Ändern der Polarität eines Signals

Wenn externe Hardware ein "active low" Signal erwartet, setzen Sie den entsprechenden *-invert* Parameter. Zum Beispiel, um das Spindelsteuersignal zu invertieren:

```
setp parport.0.pin-09-out-invert TRUE
```

Hinzufügen einer PWM-Spindeldrehzahlregelung

Wenn Ihre Spindel durch ein PWM-Signal gesteuert werden kann, verwenden Sie die Komponente „pwmgen“, um das Signal zu erzeugen:

```
loadrt pwmgen output_type=0
addf pwmgen.update servo-thread
addf pwmgen.make-pulses base-thread
net spindle-speed-cmd spindle.0.speed-out => pwmgen.0.value
net spindle-on spindle.0.on => pwmgen.0.enable
net spindle-pwm pwmgen.0.pwm => parport.0.pin-09-out
setp pwmgen.0.scale 1800 # Change to your spindle's top speed in RPM
```

Dies setzt voraus, dass die Spindelsteuerung einfach auf PWM reagiert: 0 % PWM ergibt 0 U/min, 10 % PWM ergibt 180 U/min usw. Wenn eine Mindest-PWM erforderlich ist, um die Spindel zum Drehen zu bringen, folgen Sie dem Beispiel in der Beispielkonfiguration *nist-lathe* und verwenden Sie eine *scale* Komponente.

Hinzufügen eines Aktivierungssignals (engl. enable)

Einige Verstärker (Antriebe) benötigen ein Freigabesignal, bevor sie die Bewegung der Motoren akzeptieren und befehlen. Aus diesem Grund gibt es bereits definierte Signale namens *Xen*, *Yen*, *Zen*.

Um sie zu verbinden, verwenden Sie das folgende Beispiel:

```
net Xen parport.0.pin-08-out
```

Sie können entweder einen einzigen Pin haben, der alle Antriebe aktiviert, oder mehrere, je nach Ihrer Konfiguration. Beachten Sie jedoch, dass bei einer Störung einer Achse in der Regel auch alle anderen Antriebe deaktiviert werden, so dass nur ein Freigabesignal / Pin für alle Antriebe eine gängige Praxis ist.

Externe NOTAUS (engl, ESTOP)-Taste

Die Datei standard_pinout.hal geht davon aus, dass keine externe ESTOP-Taste vorhanden ist. Weitere Informationen über einen externen Not-Aus-Schalter finden Sie in der Manpage estop_latch.

4.10. Schrittmotor Diagnostik

Wenn das, was Sie bekommen, nicht das ist, was Sie erwarten, haben Sie oft nur eine Erfahrung gemacht. Wenn man aus den Erfahrungen lernt, versteht man das Ganze besser. Die Diagnose von Problemen erfolgt am besten durch "Teilen und Herrschen". Damit ist gemeint, dass sich das Problem am schnellsten finden lässt, wenn man jedes Mal 1/2 der Variablen aus der Gleichung entfernen kann. In der realen Welt ist dies nicht immer der Fall, aber es ist normalerweise ein guter Ausgangspunkt.

4.10.1. Häufige Probleme

Stepper bewegt sich einen Schritt

Der häufigste Grund, warum sich ein Schrittmotor bei einer Neuinstallation nicht bewegt, ist, dass die Schritt- und Richtungssignale vertauscht sind. Wenn Sie die Tasten "Tippen vorwärts" und "Tippen rückwärts" abwechselnd drücken und der Schrittmotor sich jedes Mal um einen Schritt und in dieselbe Richtung bewegt, haben Sie einen Anhaltspunkt.

Keine Stepper bewegen sich

Viele Laufwerke haben einen Freigabe-Pin oder benötigen eine Ladungspumpe, um den Ausgang zu aktivieren.

Abstand nicht korrekt

Wenn Sie der Achse befehlen, sich um eine bestimmte Strecke zu bewegen, und sie sich nicht um diese Strecke bewegt, dann ist Ihre Maßstabseinstellung falsch.

4.10.2. Fehlermeldungen

Folgender Fehler

Das Konzept des Schleppfehlers ist seltsam, wenn es um Schrittmotoren geht. Da sie ein Open-Loop-System sind, gibt es keine Positionsrückmeldung, um Sie wissen zu lassen, wenn Sie tatsächlich außerhalb des Bereichs sind. LinuxCNC berechnet, ob es mit der Bewegung mithalten kann, und wenn nicht, dann gibt es einen der folgenden Fehler. Folgende Fehler sind in der Regel das Ergebnis einer der folgenden auf Stepper-Systeme.

- FERROR zu klein (engl. FERROR too small)
- MIN_FERROR zu klein (engl. MIN_FERROR too small)
- MAX_VELOCITY zu schnell (engl. MAX_VELOCITY too fast)
- MAX_ACCELERATION zu schnell (engl. MAX_ACCELERATION too fast)
- BASE_PERIOD zu lang eingestellt (engl. BASE_PERIOD set too long)
- Zu einer Achse hinzugefügtes Umkehrspiel (engl. Backlash added to an axis)

Jeder der oben genannten Punkte kann dazu führen, dass das Echtzeit-Pulsing nicht in der Lage ist, die geforderte Schrittrate einzuhalten. Dies kann passieren, wenn Sie den Latenztest nicht lange genug durchgeführt haben, um einen guten Wert für den StepConf Wizard zu erhalten, oder wenn Sie die maximale Geschwindigkeit oder die maximale Beschleunigung zu hoch eingestellt haben.

Wenn Sie Umkehrspiel hinzufügten, müssen Sie die STEPGEN_MAXACCEL bis zu doppelt so hoch wie die MAX_ACCELERATION in dem AXIS Abschnitt der INI-Datei setzen für jede Achse, für die Sie ein Umkehrspiel erhöhten. LinuxCNC verwendet "zusätzliche Beschleunigung" bei Richtungswechsel, um das Umkehrspiel zu kompensieren. Ohne die Spiel-Korrektur kann die Beschleunigung des Schritt-Generators nur ein paar Prozent über der des Bewegungsplaners liegen.

RTAPI-Fehler

Wenn Sie diese Fehlermeldung erhalten:

```
RTAPI: ERROR: Unerwartete Echtzeitverzögerung bei Aufgabe n (engl. Unexpected realtime
delay on task n)
```

Dieser Fehler wird von rtapi auf der Grundlage eines Hinweises von RTAI erzeugt, dass eine Frist verpasst wurde. Dies ist in der Regel ein Hinweis darauf, dass die BASE_PERIOD im Abschnitt [EMCMOT] der ini-Datei zu niedrig eingestellt ist. Sie sollten den Latenztest über einen längeren Zeitraum durchführen, um festzustellen, ob bei Ihnen Verzögerungen auftreten, die dieses Problem verursachen könnten. Wenn Sie den StepConf-Assistenten verwendet haben, führen Sie ihn erneut aus, testen Sie den Basisperioden-Jitter erneut und passen Sie den maximalen Basisperioden-Jitter auf der Seite mit den grundlegenden Maschineninformationen an. Möglicherweise müssen Sie den Test über einen längeren Zeitraum laufen lassen, um herauszufinden, ob eine bestimmte Hardware intermittierende Probleme verursacht.

LinuxCNC verfolgt die Anzahl der CPU-Zyklen zwischen den Aufrufen des Echtzeit-Threads. Wenn ein Element Ihrer Hardware verursacht Verzögerungen oder Ihre Echtzeit-Threads zu schnell eingestellt sind, werden Sie diesen Fehler erhalten.

NOTE Dieser Fehler wird nur einmal pro Sitzung angezeigt. Wenn Sie Ihre BASE_PERIOD zu niedrig angesetzt haben, könnten Sie Hunderttausende von Fehlermeldungen pro Sekunde erhalten, wenn mehr als eine angezeigt würde.

4.10.3. Testen

Schritt-Timing

Wenn Sie feststellen, dass eine Achse über mehrere Bewegungen hinweg an der falschen Stelle landet, ist es wahrscheinlich, dass Sie die Richtungshaltezeiten oder das Schritt-Timing für Ihre Stepper-Treiber nicht korrekt eingestellt haben. Bei jedem Richtungswechsel kann ein Schritt oder mehr verloren gehen. Wenn die Motoren blockieren, ist es auch möglich, dass Sie entweder die MAX_ACCELERATION oder MAX_VELOCITY für diese Achse zu hoch eingestellt haben.

Mit dem folgenden Programm wird die Konfiguration der Z-Achse auf ihre korrekte Einstellung geprüft. Kopieren Sie das Programm in Ihr Verzeichnis \~/emc2/nc_files und nennen Sie es TestZ.ngc oder ähnlich. Nullen Sie Ihre Maschine mit Z = 0,000 auf der Tischplatte. Laden Sie das Programm und führen Sie es aus. Es wird 200 Bewegungen von 0,5 bis 1" machen. Wenn Sie ein Konfigurationsproblem haben, werden Sie feststellen, dass die Endposition nicht bei 0,500" endet, wie es das Achsenfenster anzeigt. Um eine andere Achse zu testen, ersetzen Sie einfach die Z-Achse durch die gewünschte Achse in den G0-Zeilen.

```
( Testprogramm, um zu sehen, ob die Z-Achse ihre Position verliert )
( msg, Test 1 der Z-Achsenkonfiguration )
G20 #1000=100 ( iteriere 100 mal )
( diese Schleife hat Verzögerungen nach den Bewegungen )
( testet Beschleunigungs- und Geschwindigkeitseinstellungen )
```

```
o100 while [#1000]
   G0 Z1.000
   G4 P0.250
   G0 Z0.500
   G4 P0.250
   #1000 = [#1000 - 1]
o100 endwhile
( msg, Test 2 der Z-Achsenkonfiguration S zum Fortfahren)
M1 (hier anhalten)
#1000=100 ( Schleife 100 mal )
( die nächste Schleife hat keine Verzögerungen nach den Bewegungen )
( testet die Richtungshaltezeiten in der Treiberkonfiguration und auch die maximale
Beschleunigungseinstellung )
o101 while [#1000]
   G0 Z1.000
   G0 Z0.500
   #1000 = [#1000 - 1]
o101 endwhile
( msg, Done...Z sollte genau .5" über dem Tisch liegen )
M2
```

4.11. Filter-Programme

4.11.1. Einführung

Die meisten Bildschirme von LinuxCNC haben die Möglichkeit, geladene Dateien durch ein "Filterprogramm" zu senden oder das Filterprogramm zu verwenden, um G-Code zu machen. Ein solcher Filter kann jede gewünschte Aufgabe erledigen: Etwas so Einfaches wie sicherzustellen, dass die Datei mit *M2* endet, oder etwas so Kompliziertes wie die Erzeugung von G-Code aus einem Bild.

4.11.2. Einrichten der INI für Programmfilter

Der Abschnitt [FILTER] der INI-Datei steuert, wie die Filter funktionieren. Schreiben Sie zunächst für jeden Dateityp eine PROGRAM_EXTENSION-Zeile. Dann geben Sie das Programm an, das für jeden Dateityp ausgeführt werden soll. Dieses Programm erhält den Namen der Eingabedatei als erstes Argument und muss rs274ngc-Code in die Standardausgabe schreiben. Diese Ausgabe ist das, was im Textbereich angezeigt wird, in der Vorschau im Anzeigebereich, und dann auch von LinuxCNC ausgeführt wird. Die folgenden Zeilen fügen Unterstützung für den in LinuxCNC enthaltenen "image-to-gcode" (engl. für Bild zu G-Code) -Konverter hinzu:

```
[FILTER]
PROGRAM_EXTENSION = .png,.gif Greyscale Depth Image
png = image-to-gcode
gif = image-to-gcode
```

Es ist auch möglich, einen Interpreter anzugeben:

```
PROGRAM_EXTENSION = .py Python Script
py = python
```

Auf diese Weise kann jedes Python-Skript geöffnet werden, und seine Ausgabe wird als G-Code behandelt. Ein solches Beispielskript ist unter "nc_files/holecircle.py" verfügbar. Dieses Skript erzeugt G-Code für das Bohren einer Reihe von Löchern entlang des Umfangs eines Kreises.

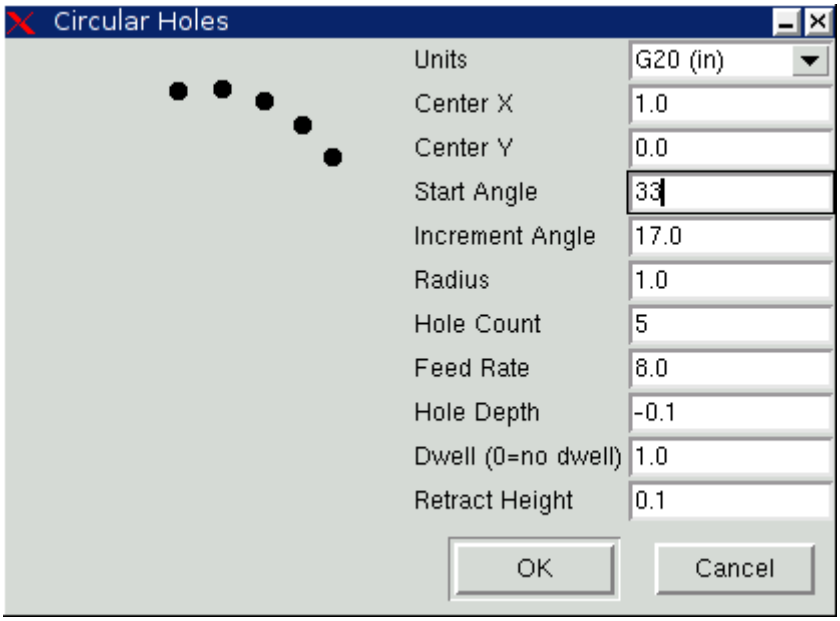


Figure 62. Kreisförmige Löcher

Wenn das Filterprogramm Zeilen in der folgenden Form an stderr sendet:

```
FILTER_PROGRESS=10
```

Sie setzt den Fortschrittsbalken des Bildschirms auf den angegebenen Prozentsatz (in diesem Fall 10). Diese Funktion sollte von jedem Filter verwendet werden, der lange läuft.

4.11.3. Erstellung von Filterprogrammen auf Python-Basis

Hier ist ein sehr einfaches Beispiel für die Filtermechanik: Wenn ein Linucnc-Bildschirm, der Programmfilterung bietet, durchläuft, wird jede 100stel Sekunde eine Zeile G-Code erzeugt und auf die Standardausgabe geschrieben. Außerdem sendet es eine Fortschrittsmeldung an den UNIX-Standardfehler-Ausgabe. Wenn ein Fehler auftritt, gibt es eine Fehlermeldung aus und beendet sich mit dem Exitcode 1.

```
import time
import sys

for i in range(0,100):
    try:
        # Rechenzeit simulieren
        time.sleep(.1)

        # Ausgabe einer Zeile G-Code
        print('G0 X1', file=sys.stdout)

        # Fortschritt aktualisieren
        print('FILTER_PROGRESS={}'.format(i), file=sys.stderr)
```

```
    except:
        # Dies führt zu einer Fehlermeldung
        print('Fehler; Aber das war nur ein Test', file=sys.stderr)
        raise SystemExit(1)
```

Hier ist ein ähnliches Programm, aber es kann tatsächlich filtern. Es zeigt einen PyQt5-Dialog mit einer Abbruch-Schaltfläche an. Dann liest es das Programm Zeile für Zeile und gibt es an die Standardausgabe weiter. Während es weiterläuft, aktualisiert es jeden Prozess, der auf die Standardfehlerausgabe hört.

```
#!/usr/bin/env python3

import sys
import os
import time

from PyQt5.QtWidgets import (QApplication, QDialog, QDialogButtonBox,
                             QVBoxLayout,QDialogButtonBox)
from PyQt5.QtCore import QTimer, Qt

class CustomDialog(QDialog):

    def __init__(self, path):
        super(CustomDialog, self).__init__(None)
        self.setWindowFlags(self.windowFlags() | Qt.WindowStaysOnTopHint)
        self.setWindowTitle("Filter-with-GUI Test")

        QBtn = QDialogButtonBox.Cancel

        self.buttonBox = QDialogButtonBox(QBtn)
        self.buttonBox.rejected.connect(self.reject)

        self.layout = QVBoxLayout()
        self.layout.addWidget(self.buttonBox)
        self.setLayout(self.layout)

        self.line = 0
        self._percentDone = 0

        if not os.path.exists(path):
            print("Path: '{}' existiert nicht:".format(path), file=sys.stderr)
            raise SystemExit(1)

        self.infile = open(path, "r")
        self.temp = self.infile.readlines()

        # calculate percent update interval
        self.bump = 100/float(len(self.temp))

        self._timer = QTimer()
        self._timer.timeout.connect(self.process)
        self._timer.start(100)

    def reject(self):
        # This provides an error message
        print('You asked to cancel before finished.', file=sys.stderr)
```

```
        raise SystemExit(1)

    def process(self):
        try:
            # nächste Codezeile erhalten
            codeLine = self.temp[self.line]

            # die Zeile irgendwie verarbeiten

            # Verarbeiteten Code ausgeben
            print(codeLine, file=sys.stdout)
            self.line +=1

            # update progress
            self._percentDone += self.bump
            print('FILTER_PROGRESS={}'.format(int(self._percentDone)), file=sys.stderr)

            # if done Ende ohne Fehler/Fehlermeldung
            if self._percentDone >= 99:
                print('FILTER_PROGRESS=-1', file=sys.stderr)
                self.infile.close()
                raise SystemExit(0)

        except Exception as e:
            # Dies liefert eine Fehlermeldung
            print(('Something bad happened:',e), file=sys.stderr)
            # dies signalisiert, dass die Fehlermeldung angezeigt werden soll
            raise SystemExit(1)

if __name__ == "__main__":
    if (len(sys.argv)>1):
        path = sys.argv[1]
    else:
        path = None
    app = QApplication(sys.argv)
    w = CustomDialog(path=path)
    w.show()
    sys.exit( app.exec_() )
```

[1] Dieser Abschnitt bezieht sich auf die Verwendung **stepgen**, LinuxCNCs eingebauten Schritt-Generator. Einige Hardware-Geräte haben ihre eigenen Schritt-Generator und nicht mit LinuxCNC ' s built-in ein. In diesem Fall, verweisen wir auf Ihr Hardware-Handbuch

[2] dirhold bezieht sich auf einen Parameter, der die Länge der Richtungshaltezeit einstellt.

[3] Der schnellste Thread im LinuxCNC-Setup, normalerweise wird der Code alle paar zehn Mikrosekunden ausgeführt.

Chapter 5. HAL (Hardware Abstraction Layer)

5.1. HAL Einführung

Bei LinuxCNC geht es um die Interaktion mit Hardware. Aber nur wenige Benutzer haben die gleichen genauen Hardware-Spezifikationen - ähnlich, aber nicht gleich. Und selbst für die exakt gleiche Hardware, kann es verschiedene Möglichkeiten, diese zu benutzen, sagen wir für verschiedene Materialien oder mit verschiedenen Fräsen, die Anpassungen an der Steuerung eines bereits laufenden Systems erfordern würde. Es wurde eine Abstraktion benötigt, um die Konfiguration von LinuxCNC für eine Vielzahl von Hardware-Geräten zu erleichtern. Auf der höchsten Ebene könnte es einfach eine Möglichkeit sein, eine Reihe von "Bausteinen" zu laden und miteinander zu verbinden, um ein komplexes System zusammenzustellen.

Dieses Kapitel führt Sie in die Hardware-Abstraktionsschicht ein. Sie werden sehen, dass viele der Bausteine tatsächlich Treiber für Hardwaregeräte sind. HAL kann jedoch mehr als nur Hardwaretreiber zu konfigurieren.

5.1.1. HAL Übersicht

Die Hardware-Abstraktionsschicht (oder mit einem Verweis auf den 2001 Space Odyssey Film einfach "HAL") ist eine Software zur

- stellen die Infrastruktur für die Kommunikation mit und zwischen den zahlreichen Software- und Hardwarekomponenten des Systems bereit.
- diese Informationen bei ihrem Fluss von Komponente zu Komponente optional verarbeiten und/oder überschreiben.

An sich ist diese Middleware bezüglich seiner Anwendung auf CNC nicht speziell ausgelegt. Eine Internetsuche fand beispielsweise eine astronomische Anwendung zur Steuerung von Teleskopen mit LinuxCNC. Motoren bewegen das Teleskop in die richtige Position, und man muss wissen, wie man die Motoraktivität mit den Auswirkungen dieser Positionierung auf die reale Welt abbildet. Eine solche Synchronisierung der Motorpositionen mit den Positionen in der realen Welt erinnert an das, was CNC-Maschinen tun müssen, oder gar ein Raumschiff.

Jede Maschinensteuerung muss kennen:

- über ihren internen Zustand und dessen Abbildung auf die Umgebung (Maschinenkoordinaten, Zustand von Schaltern/Reglern),
- wie die Aktoren diesen Zustand verändern sollen,
- wie die Aktualisierung des internen Zustands durch Sensoren (Encoder, Sonden) ermöglichen.

Die HAL-Schicht besteht aus Teilen (als „Komponenten“ bezeichnet), die

- miteinander verbunden sind, z. B. um die Positionsdaten zu aktualisieren oder um den Planungsalgorithmus den Motoren den nächsten Schritt mitteilen zu lassen.
- wissen vielleicht, wie man mit Hardware kommuniziert,

- kann einfach eingehende Daten verarbeiten und Datenausgaben für andere Komponenten bereitstellen,
- werden immer periodisch ausgeführt
  - mit einer sehr hohen Frequenz von wenigen Mikrosekunden (µs) Ausführungszeit, genannt Basis-Thread, z.B. um
    1. einen Schrittmotor dazu zu bewegen, einen weiteren Schritt zu machen
    2. die von einem Encoder angezeigte Position auslesen.
  - mit einer niedrigeren Frequenz pro Millisekunde (ms), z.B. um
    1. die Planung für die nächsten Züge anzupassen, um eine G-Code-Anweisung auszuführen.
  - als Nicht-Echtzeit-"User-Space"-Komponenten, die wie jede andere Software eine "Hauptschleife" (engl. main loop) ausführen und unterbrochen oder verzögert werden können, wenn der Rest des Systems ausgelastet oder überlastet ist.

Zusammengenommen ermöglicht HAL

1. für eine Maschine zu programmieren, die der Programmierer nicht direkt kennt, sich aber auf eine Programmierschnittstelle mit genau spezifizierter Wirkung auf die Maschine verlassen kann. Diese Schnittstelle kann verwendet werden, um
   - der Maschine sagen, was sie tun soll
   - zu hören, was die Maschine über den Zustand sagen will, in dem sie gerade ist.
2. Vertikale Abstraktionen: Der menschliche Systemintegrator einer solchen Maschine verwendet HAL
   - um zu beschreiben, wie die Maschine aussieht und wie welches Kabel welchen Motor steuert, der welche Achse antreibt.
   - Die Beschreibung der Maschine, die Schnittstellen des Programmierers und die Benutzerschnittstelle "treffen" sich irgendwie in dieser abstrakten Schicht.
3. Horizontale Abstraktionen:
   - Nicht alle Maschinen verfügen über alle möglichen Funktionen
   - Mühlen, Drehmaschinen und Roboter teilen viele
     - Merkmale (Motoren, Gelenke, ...),
     - Planungsalgorithmen für ihre Bewegungen.

HAL hat keine direkte Interaktion mit dem Benutzer. Es wurden jedoch mehrere Schnittstellen bereitgestellt, mit denen HAL manipuliert werden kann

- über die Kommandozeile mit dem Befehl "halcmd".
- aus Python-Skripten und
- aus C/C++-Programmen heraus,

aber keine dieser Schnittstellen ist "HAL selbst".

HAL selbst ist kein Programm, es besteht aus einer oder mehreren Listen geladener Programme (die

Komponenten), die periodisch ausgeführt werden (in strenger Reihenfolge), und einem Bereich des gemeinsamen Speichers für den Datenaustausch verwenden. Das Haupt-HAL-Skript läuft nur einmal beim Maschinenstart, die Einrichtung der Echtzeit-Gewinde und der gemeinsamen Speicherplätze, das Laden der Komponenten und die Einrichtung der Datenverbindungen zwischen ihnen (die "Signale" und "Pins").

Im Prinzip könnten sich mehrere Maschinen eine gemeinsame HAL teilen, um ihnen die Zusammenarbeit zu ermöglichen, jedoch ist die aktuelle Implementierung von LinuxCNC auf einen einzigen Interpreter und ein einzelnes Task-Modul beschränkt. Derzeit handelt es sich dabei fast immer um einen G-Code-Interpreter und den "milltask" (der sich auch für Drehmaschinen und für Roboter bewährt hat), aber diese Module sind zur Ladezeit auswählbar. Mit einem zunehmenden Interesse an der Steuerung mehrerer kooperierender Maschinen ist die Überwindung dieser Einschränkung wahrscheinlich einer der wichtigsten Schritte für die zukünftige Entwicklung von LinuxCNC. Es ist jedoch ein bisschen knifflig und die Community ist immer noch dabei, ihre Gedanken dazu zu organisieren.

HAL ist das Herzstück von LinuxCNC und wird von allen Teilen von LinuxCNC, einschließlich der GUIs, verwendet und/oder erweitert. Der G-Code-Interpreter (oder alternative Sprache) weiß, wie der G-Code zu interpretieren ist, und übersetzt ihn in Maschinenoperationen, indem er Signale in HAL auslöst. Der Benutzer kann HAL auf verschiedene Weise abfragen, um Informationen über seinen Zustand zu erhalten, der dann auch den Zustand des Computers darstellt. Während des Schreibens während der Entwicklung von Version 2.9 machen die GUIs immer noch eine kleine Ausnahme von dieser Regel und wissen möglicherweise etwas, das HAL nicht weiß (und auch nicht wissen muss).

5.1.2. Kommunikation

HAL ist besonders, da er richtig schnell kommunizieren kann

- mit anderen Programmen, vor allem aber
- mit seinen Komponenten, die in der Regel in einem der Echtzeit-Threads ausgeführt werden.

Und während der Kommunikation muss sich der Teil von LinuxCNC, mit dem gesprochen wird, nicht auf die Kommunikation vorbereiten: Alle diese Aktionen werden asynchron ausgeführt, d.h. keine Komponente unterbricht ihre reguläre Ausführung, um ein Signal zu empfangen, und Signale können sofort gesendet werden, d.h. eine Anwendung kann warten, bis eine bestimmte Nachricht angekommen ist - wie ein Enable-Signal, sie muss sich jedoch nicht auf den Empfang dieser Nachricht vorbereiten.

Das Kommunikationssystem

- repräsentiert und steuert die gesamte Hardware, die an das System angeschlossen ist,
- startet und stoppt andere kommunizierende Programme.

Die Kommunikation mit der Hardware der Maschine selbst erfolgt durch jeweilige individuell darauf ausgelegte HAL-Komponenten.

The HAL layer is a shared space in which all the many parts that constitute LinuxCNC are exchanging information. That space features pins that are identified by a name, though a LinuxCNC engineer may prefer the association with a pin of an electronic circuit. These pins can carry numerical and logical

values, boolean, floating point and signed and unsigned integers. There is also a pin type named hal_port intended for byte streams, and a framework for exchanging more complex data called hal_stream (which uses a private shared memory area, rather than a HAL pin). These latter two types are used relatively infrequently.

Mit HAL können Sie ein Signal an diesen benannten Pin senden. Jeder Teil von HAL kann den Pin lesen, der dann diesen zuvor gesendeten Signalwert hält. Dies gilt so lange, bis ein neues Signal an diesen Pin gesendet wird, um den vorherigen Wert zu ersetzen. Das zentrale Nachrichtenaustauschsystem von HAL ist CNC-unabhängig, HAL wird jedoch mit einer großen Anzahl von Komponenten ausgeliefert, die viel über CNC wissen und diese Informationen über Pins präsentieren. Es gibt Pins, die darstellen

- statische (unveränderliche) Informationen über die Maschine
- den aktuellen Zustand der Maschine
  - Endschalter (engl. end switches)
  - Positionen, die von Schrittmotoren gezählt oder von Encodern gemessen werden
- Empfänger von Anweisungen
  - manuelle Steuerung der Maschinenposition ("Jogging")
  - Positionen, die Schrittmotoren als nächstes einnehmen sollten

In Analogie zu elektronischen Kabeln können Pins verdrahtet werden, sodass die Wertänderung an einem Pin als Eingabe für einen anderen Pin dient. HAL-Komponenten bereiten solche Ein- und Ausgangspins vor und werden so automatisch zur Ausführung veranlasst.

HAL-Komponenten

Die vielen "Experten"-Softwareteile von LinuxCNC werden typischerweise als *Komponenten* von HAL implementiert, konzeptionell auch als *Module* bezeichnet. Diese computerimplementierten Experten lesen ständig von HAL über einen Zustand, den die Maschine anstreben sollte, und vergleichen diesen gewünschten Zustand mit dem Zustand, in dem sich die Maschine gerade befindet. Wenn es einen Unterschied zwischen dem, was sein sollte, und dem, was der aktuelle Zustand ist, gibt, werden Maßnahmen ergriffen, um diesen Unterschied zu verringern, während ständig Aktualisierungen der aktuellen Zustände zurück in den HAL-Datenraum geschrieben werden.

Es gibt Komponenten, die darauf spezialisiert sind, mit Schrittmotoren zu kommunizieren, und andere Komponenten, die wissen, wie man Servos steuert. Auf einer höheren Ebene wissen einige Komponenten, wie die Achsen der Maschine in 3D angeordnet sind, und wieder andere, wie sie eine gleichmäßige Bewegung von einem Punkt im Raum zum anderen ausführen können. Drehmaschinen, Fräsen und Roboter unterscheiden sich in den LinuxCNC-Komponenten, die gerade aktiv sind, d.h. die von einer HAL-Konfigurationsdatei für diese Maschine geladen wird. Dennoch mögen zwei Maschinen sehr unterschiedlich aussehen, da sie für sehr unterschiedliche Zwecke gebaut wurden, aber wenn beide Servomotoren verwenden, können sie immer noch beide die gleiche HAL-Servokomponente verwenden.

Ursprung des Strebens sich zu bewegen

Auf der niedrigsten (nahe an Hardware) Ebene, z.B. für Schrittmotoren, ist die Beschreibung eines Zustands dieses Motors sehr intuitiv: Es ist die Anzahl der Schritte in einer bestimmten Richtung. Ein Unterschied zwischen der Sollposition und der Istposition führt zu einer Bewegung. Geschwindigkeiten,

Beschleunigungen und andere Parameter können im Bauteil selbst intern begrenzt sein oder gegebenenfalls durch vorgeschaltete Komponenten begrenzt sein. (In den meisten Fällen wurden z.B. die an die Schrittgeneratorkomponenten gesendeten Moment-zu-Moment-Achsen-Positionswerte bereits begrenzt und entsprechend den konfigurierten Maschinengrenzen bzw. der aktuellen Vorschubrate angepasst.)

Jede G-Code-Linie wird interpretiert und löst eine Reihe von Routinen aus, die wiederum wissen, wie mit Komponenten einer mittleren Schicht kommunizieren, z.B. um auf einem Kreis zu fahren.

Pins und Signale

HAL hat einen besonderen Platz im Herzen seiner Programmierer für die Darstellung des Datenflusses zwischen Modulen. Wenn traditionelle Programmierer an Variablen, Adressen oder I/O-Ports denken, bezieht sich HAL auf "Pins". Und diese Pins (deutsch auch "Stifte") sind über Signale mit oder mit Werten verbunden. Ähnlich wie ein Elektroingenieur Drähte zwischen Pins von Bauteilen einer Mühle verbinden würde, stellt ein HAL-Ingenieur den Datenfluss zwischen Pins von Modulinstanzen her.

Die LinuxCNC GUIS (AXIS, GMOCCAPY, Touchy, etc.) werden die Zustände einiger Pins (wie Limit Switches) darstellen, aber auch andere grafische Tools zur Fehlerbehebung und Konfiguration existieren: Halshow, Halmeter, Halscope und Halreport.

Der Rest dieser Einleitung präsentiert

- die Syntax, wie Pins verschiedener Komponenten in den HAL-Konfigurationsdateien verbunden sind, und
- Software zur Überprüfung der Werte von Pins
  - zu jedem Zeitpunkt,
  - sich im Laufe der Zeit entwickeln.

5.1.3. HAL System Design

.HAL basiert auf traditionellen Systementwurfstechniken.

HAL basiert auf denselben Prinzipien, die auch bei der Entwicklung von Hardware-Schaltungen und -Systemen zum Einsatz kommen, weshalb es sinnvoll ist, zunächst diese Prinzipien zu untersuchen. Jedes System, einschließlich einer CNC-Maschine, besteht aus miteinander verbundenen Komponenten. Bei einer CNC-Maschine könnten diese Komponenten die Hauptsteuerung, Servoverstärker oder Schrittmotoren, Motoren, Encoder, Endschalter, Drucktastenschalter, vielleicht ein VFD für den Spindelantrieb, eine SPS zur Steuerung eines Werkzeugwechslers usw. sein. Der Maschinenbauer muss diese Teile auswählen, montieren und miteinander verdrahten, um ein komplettes System zu erhalten.

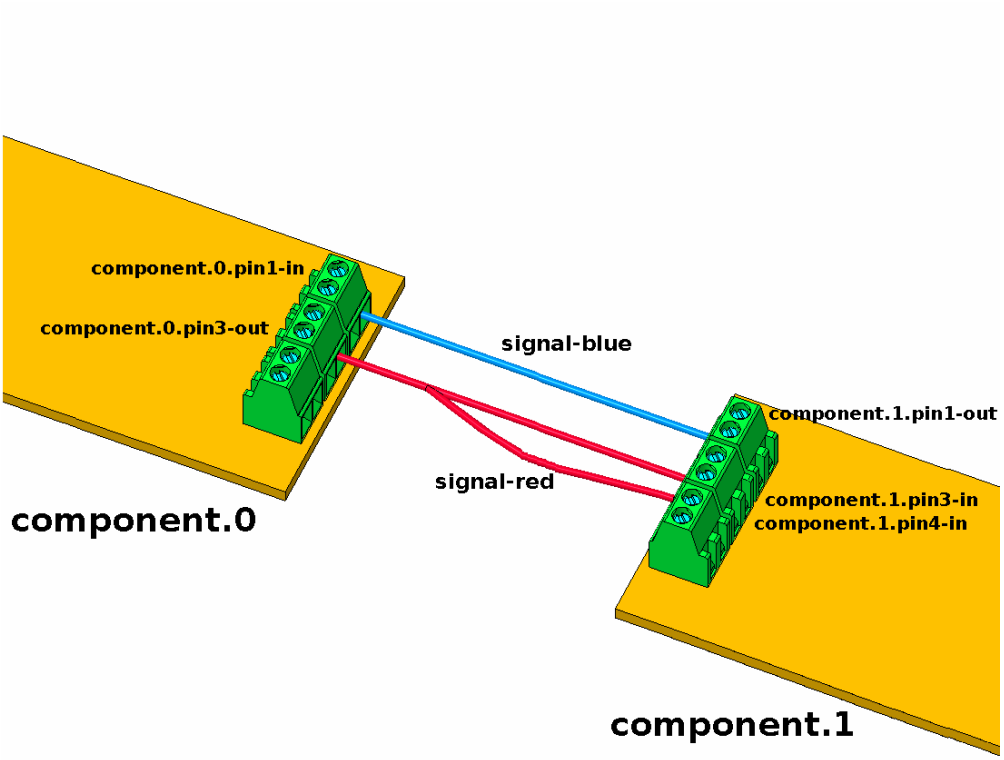


Figure 63. HAL-Konzept - Verbinden wie elektrische Schaltkreise.

Abbildung 1 würde wie folgt in HAL-Code geschrieben:

```
net signal-blue    component.0.pin1-in      component.1.pin1-out
net signal-red     component.0.pin3-out     component.1.pin3-in     component.1.pin4-in
```

Werkstück Auswahl

Der Maschinenbauer muss sich nicht darum kümmern, wie jedes einzelne Teil funktioniert. Er behandelt sie als Blackboxen. In der Konstruktionsphase entscheidet er, welche Teile er verwenden will - Stepper oder Servos, welche Marke von Servoverstärker, welche Art von Endschaltern und wie viele, usw. Die Entscheidung des Integrators, welche Komponenten er verwendet, basiert auf der Funktion der jeweiligen Komponente und den vom Hersteller des Geräts angegebenen Spezifikationen. Die Größe eines Motors und die Last, die er antreiben muss, beeinflussen die Wahl des Verstärkers, der für den Betrieb benötigt wird. Die Wahl des Verstärkers kann sich auf die Art der Rückkopplung auswirken, die der Verstärker benötigt, sowie auf die Geschwindigkeits- oder Positionssignale, die von einer Steuerung an den Verstärker gesendet werden müssen.

In der HAL-Welt muss der Integrator entscheiden, welche HAL-Komponenten benötigt werden. In der Regel wird für jede Schnittstellenkarte ein Treiber benötigt. Zusätzliche Komponenten können für die Software-Generierung von Schrittimpulsen, SPS-Funktionen und eine Vielzahl anderer Aufgaben erforderlich sein.

Design der Verbindungen (engl. interconnections)

Der Konstrukteur eines Hardwaresystems wählt nicht nur die Teile aus, er entscheidet auch, wie diese Teile miteinander verbunden werden. Jeder schwarze Kasten hat Anschlüsse, vielleicht nur zwei für

einen einfachen Schalter oder Dutzende für einen Servoantrieb oder eine SPS. Sie müssen miteinander verdrahtet werden. Die Motoren werden mit den Servoverstärkern verbunden, die Endschalter mit der Steuerung und so weiter. Während der Maschinenbauer an der Konstruktion arbeitet, erstellt er einen großen Verdrahtungsplan, der zeigt, wie alle Teile miteinander verbunden werden sollen.

Bei der Verwendung von HAL werden die Komponenten durch Signale miteinander verbunden. Der Designer muss entscheiden, welche Signale benötigt werden und was sie verbinden sollen.

Implementierung

Sobald der Schaltplan fertig ist, kann die Maschine gebaut werden. Die Teile müssen beschafft und montiert werden, und dann werden sie entsprechend dem Schaltplan miteinander verbunden. In einem physischen System besteht jede Verbindung aus einem Stück Draht, das abgeschnitten und an die entsprechenden Klemmen angeschlossen werden muss.

HAL bietet eine Reihe von Werkzeugen, die beim "Aufbau" eines HAL-Systems helfen. Mit einigen dieser Werkzeuge können Sie einen einzelnen "Draht" anschließen (oder abziehen). Andere Werkzeuge ermöglichen es Ihnen, eine vollständige Liste aller Teile, Drähte und anderer Informationen über das System zu speichern, so dass es mit einem einzigen Befehl "neu aufgebaut" werden kann.

Testen

Nur sehr wenige Maschinen funktionieren beim ersten Mal richtig. Bei der Prüfung kann der Konstrukteur ein Messgerät verwenden, um zu sehen, ob ein Endschalter funktioniert, oder um die Gleichspannung an einem Servomotor zu messen. Er kann ein Oszilloskop anschließen, um die Einstellung eines Antriebs zu überprüfen oder um nach elektrischen Störungen zu suchen. Vielleicht findet er ein Problem, das eine Änderung des Schaltplans erfordert; vielleicht muss ein Teil anders angeschlossen oder durch etwas völlig anderes ersetzt werden.

HAL bietet die Software-Äquivalente eines Voltmeters, Oszilloskops, Signalgenerators und anderer Werkzeuge, die zum Testen und Abstimmen eines Systems benötigt werden. Mit denselben Befehlen, die zum Aufbau des Systems verwendet werden, können auch Änderungen vorgenommen werden.

Zusammenfassung

Dieses Dokument richtet sich an Personen, die bereits wissen, wie man diese Art von Hardware-Systemintegration durchführt, die aber nicht wissen, wie man die Hardware mit LinuxCNC verbindet. Siehe den Abschnitt Remote Start Example in der HAL UI Examples Dokumentation.

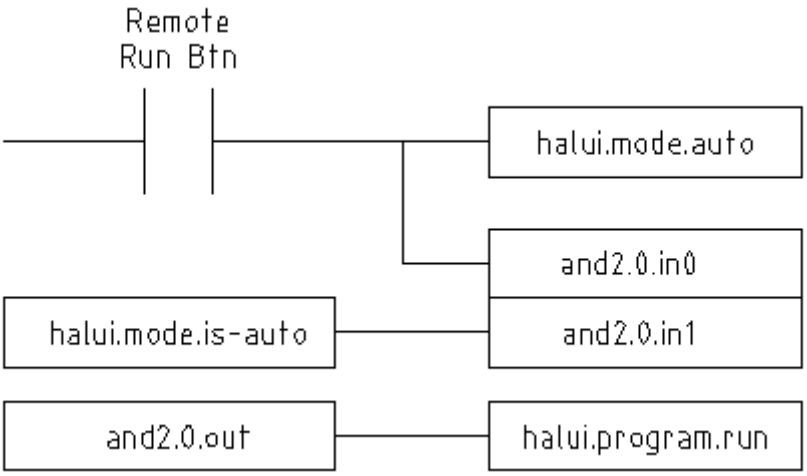


Figure 64. Remote-Start-Beispiel (Schema)

Das oben beschriebene traditionelle Hardware-Design endet am Rande der Hauptsteuerung. Außerhalb der Steuerung befinden sich eine Reihe relativ einfacher Kästen, die miteinander verbunden sind, um das zu tun, was erforderlich ist. Im Inneren ist die Steuerung ein großes Rätsel - eine riesige schwarze Box, von der wir hoffen, dass sie funktioniert.

HAL erweitert diese traditionelle Hardware-Design-Methode auf das Innere der großen Blackbox. Es macht Gerätetreiber und sogar einige interne Teile des Controllers zu kleineren Black Boxes, die miteinander verbunden und sogar ersetzt werden können, genau wie die externe Hardware. So kann der "Systemschaltplan" einen Teil des internen Steuergeräts zeigen und nicht nur eine große Blackbox. Und, was am wichtigsten ist, es ermöglicht dem Integrator, den Controller mit denselben Methoden zu testen und zu modifizieren, die er auch für den Rest der Hardware verwenden würde.

Begriffe wie Motoren, Ampere und Encoder sind den meisten Maschinenintegratoren vertraut. Wenn wir über die Verwendung eines besonders flexiblen, achtadrigen, abgeschirmten Kabels sprechen, um einen Drehgeber mit der Servo-Eingangsplatine im Computer zu verbinden, versteht der Leser sofort, worum es sich handelt, und wird zu der Frage geführt, "welche Arten von Steckern ich für die beiden Enden benötige". Die gleiche Art von Denken ist für das HAL wesentlich, aber der spezifische Gedankengang braucht vielleicht ein bisschen, um auf den richtigen Weg zu kommen. Die Verwendung von HAL-Wörtern mag anfangs etwas seltsam erscheinen, aber das Konzept, von einer Verbindung zur nächsten zu arbeiten, ist dasselbe.

Diese Idee, den Schaltplan auf das Innere des Controllers auszudehnen, ist das eigentliche Anliegen von HAL. Wenn Sie mit der Idee, Hardware-Blackboxen miteinander zu verbinden, vertraut sind, werden Sie wahrscheinlich wenig Probleme haben, HAL für die Verbindung von Software-Blackboxen zu verwenden.

5.1.4. HAL Konzepte

Dieser Abschnitt ist ein Glossar, in dem die wichtigsten HAL-Begriffe definiert werden. Er unterscheidet sich jedoch etwas von einem herkömmlichen Glossar, da die Begriffe nicht in alphabetischer Reihenfolge angeordnet sind. Sie sind nach ihrer Beziehung oder ihrem Fluss in der HAL-Welt geordnet.

Komponente:: Wenn wir über Hardware-Design sprechen, bezeichnen wir die einzelnen Teile als *Teile*, *Bausteine*, *Black Boxes* usw. Das HAL-Äquivalent ist eine *Komponente* oder *HAL-Komponente*. In diesem

Dokument wird der Begriff "HAL-Komponente" verwendet, wenn eine Verwechslung mit anderen Arten von Komponenten wahrscheinlich ist, aber normalerweise wird nur "Komponente" verwendet. Eine HAL-Komponente ist ein Stück Software mit genau definierten Eingängen, Ausgängen und Verhalten, das installiert und nach Bedarf miteinander verbunden werden kann. ++ Viele HAL-Komponenten modellieren das Verhalten eines greifbaren Teils einer Maschine, und ein **Pin** kann tatsächlich mit einem **physischen Pin** des Geräts verbunden werden, um mit ihm zu kommunizieren, daher die Namen. In den meisten Fällen ist dies jedoch nicht der Fall. Stellen Sie sich eine Nachrüstung einer manuellen Drehmaschine/Fräse vor. Was LinuxCNC implementiert, ist die Art und Weise, wie sich die Maschine der Außenwelt präsentiert, und es ist sekundär, ob die Implementierung, wie man einen Kreis zeichnet, bereits auf der Maschine implementiert ist oder von LinuxCNC bereitgestellt wird. Und es ist üblich, Knöpfe zur imaginären Nachrüstung hinzuzufügen, die eine Aktion **signalisieren**, wie ein Notaus. LinuxCNC und die Maschine werden eins. Und das ist durch den HAL.

Parameter:: Viele Hardwarekomponenten verfügen über Einstellmöglichkeiten, die nicht mit anderen Komponenten verbunden sind, auf die aber dennoch zugegriffen werden muss. Zum Beispiel haben Servoverstärker oft Trimmpotis, mit denen sie eingestellt werden können, und Testpunkte, an denen ein Messgerät oder Oszilloskop angeschlossen werden kann, um die Ergebnisse der Einstellung zu überprüfen. Auch HAL-Komponenten können solche Elemente haben, die als "Parameter" bezeichnet werden. Es gibt zwei Arten von Parametern: Eingangsparameter entsprechen Trimmpotentiometern - es handelt sich um Werte, die vom Benutzer eingestellt werden können und nach der Einstellung fest bleiben. Ausgangsparameter können nicht vom Benutzer eingestellt werden - sie entsprechen Testpunkten, mit denen interne Signale überwacht werden können.

Pin:: Hardware-Komponenten haben Anschlüsse, über die sie miteinander verbunden werden. Das HAL-Äquivalent ist ein "Pin" oder "HAL-Pin". Der Begriff "HAL-Pin" wird verwendet, wenn es nötig ist, um Verwechslungen zu vermeiden. Alle HAL-Pins sind benannt, und die Pin-Namen werden verwendet, wenn sie miteinander verbunden werden. HAL-Pins sind Software-Entitäten, die nur innerhalb des Computers existieren.

Physischer Pin (Steckkontakt):: Viele E/A-Geräte haben echte physische Pins oder Anschlüsse, die mit externer Hardware verbunden sind, z. B. die einzelnen Steckkontakte eines Parallelport-Anschlusses. Um Verwirrung zu vermeiden, werden diese als "physische Pins" (engl. physical pins) bezeichnet. Dies sind die Dinge, die in die reale Welt *hineinragen*.

NOTE Sie werden sich vielleicht fragen, welche Beziehung zwischen den HAL_pins, physical_pins und externen Elementen wie Encodern oder einer STG-Karte besteht: Wir haben es hier mit Schnittstellen vom Typ Datenübersetzung/-umwandlung zu tun.

Signal:: In einer physischen Maschine sind die Anschlüsse der realen Hardwarekomponenten durch Drähte miteinander verbunden. Das HAL-Äquivalent eines Drahtes ist ein "Signal" oder "HAL-Signal". HAL-Signale verbinden HAL-Pins miteinander, wie vom Maschinenbauer gewünscht. HAL-Signale können nach Belieben abgetrennt und wieder angeschlossen werden (sogar während die Maschine läuft).

Typ:: Bei der Verwendung echter Hardware würden Sie einen 24 Volt Relaisausgang nicht an den +/-10 V Analogeingang eines Servoverstärkers anschließen. Für HAL-Pins gelten die gleichen Einschränkungen, die auf ihrem Typ basieren. Sowohl Pins als auch Signale haben Typen, und Signale können nur an Pins desselben Typs angeschlossen werden. Derzeit gibt es 4 Typen, wie folgt:

+ - bool - a single TRUE/FALSE or ON/OFF value - real - a double precision floating point value, with approximately 53 bits of resolution and over 1000 bits of dynamic range. - uint - a 64 bit unsigned integer, legal values are 0 to 18,446,744,073,709,551,615 - sint - a 64 bit signed integer, legal values are -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807

Funktion:: Echte Hardwarekomponenten reagieren in der Regel sofort auf ihre Eingänge. Wenn sich beispielsweise die Eingangsspannung eines Servoverstärkers ändert, dann ändert sich automatisch auch der Ausgang. Softwarekomponenten können jedoch nicht "automatisch" handeln. Jede Komponente verfügt über einen spezifischen Code, der ausgeführt werden muss, um das zu tun, was die Komponente tun soll. In einigen Fällen wird dieser Code einfach als Teil der Komponente ausgeführt. In den meisten Fällen jedoch, insbesondere bei Echtzeitkomponenten, muss der Code in einer bestimmten Reihenfolge und in bestimmten Abständen ausgeführt werden. So sollten beispielsweise Eingaben gelesen werden, bevor Berechnungen mit den Eingabedaten durchgeführt werden, und Ausgaben sollten erst dann geschrieben werden, wenn die Berechnungen abgeschlossen sind. In diesen Fällen wird der Code dem System in Form von einer oder mehreren "Funktionen" zur Verfügung gestellt. Jede Funktion ist ein Codeblock, der eine bestimmte Aktion ausführt. Der Systemintegrator kann "Threads" verwenden, um eine Reihe von Funktionen zu planen, die in einer bestimmten Reihenfolge und in bestimmten Zeitabständen ausgeführt werden sollen.

Thread:: Ein "Thread" ist eine Liste von Funktionen, die in bestimmten Zeitabständen als Teil einer Echtzeitaufgabe ausgeführt werden. Wenn ein Thread zum ersten Mal erstellt wird, hat er ein bestimmtes Zeitintervall (Periode), aber keine Funktionen. Funktionen können dem Thread hinzugefügt werden, um diese bei jeder Ausführung des Threads der Reihe nach auch auszuführen.

Angenommen, wir haben eine Parport-Komponente mit dem Namen hal_parport. Diese Komponente definiert einen oder mehrere HAL-Pins für jeden physischen Pin. Die Pins werden im Dokumentabschnitt dieser Komponente beschrieben: Ihre Namen, wie sich jeder Pin auf den physischen Pin bezieht, ob sie invertiert sind, ob Sie die Polarität ändern können usw. Aber das allein bringt die Daten nicht von den HAL-Pins zu den physischen Pins. Es braucht Code, um das zu tun, und hier kommen Funktionen ins Spiel. Die Parport-Komponente benötigt mindestens zwei Funktionen: Eine, um die physischen Eingangspins zu lesen und die HAL-Pins zu aktualisieren, die andere, um Daten von den HAL-Pins zu nehmen und sie auf die physischen Ausgangspins zu schreiben. Beide Funktionen sind Teil des parport-Treibers.

5.1.5. HAL-Komponenten

Jede HAL-Komponente ist ein Stück Software mit genau definierten Eingängen, Ausgängen und Verhaltensweisen, das installiert und nach Bedarf miteinander verbunden werden kann. Der Abschnitt HAL Components List listet alle verfügbaren Komponenten und eine kurze Beschreibung ihrer Funktionen auf.

5.1.6. Timing-Probleme in HAL

Im Gegensatz zu den physikalischen Verdrahtungsmodellen zwischen Black Boxes, auf denen HAL, wie wir gesagt haben, basiert, reicht das einfache Verbinden zweier Pins mit einem HAL-Signal bei weitem nicht aus, um die Wirkung des physikalischen Falles zu erreichen.

Echte Relaislogik besteht aus miteinander verbundenen Relais, und wenn sich ein Kontakt öffnet oder

schließt, fließt (oder stoppt) sofort Strom. Andere Spulen können ihren Zustand ändern usw., und das alles "passiert". In der SPS-Kontaktplanlogik funktioniert das jedoch nicht so. In der Regel wird in einem einzigen Durchlauf durch den Kontaktplan jede Sprosse in der Reihenfolge ausgewertet, in der sie erscheint, und zwar nur einmal pro Durchlauf. Ein perfektes Beispiel ist ein Kontaktplan mit einem Öffnerkontakt in Reihe mit einer Spule. Der Kontakt und die Spule gehören zum selben Relais.

Wäre dies ein herkömmliches Relais, würden sich die Kontakte öffnen, sobald die Spule erregt ist, und die Spule wieder abschalten. Das heißt, die Kontakte schließen sich wieder, usw. Das Relais wird zu einem Summton.

Wenn bei einer SPS die Spule ausgeschaltet und der Kontakt geschlossen ist, wenn die SPS mit der Auswertung des Strompfads beginnt, dann ist die Spule eingeschaltet, wenn sie diesen Durchgang beendet hat. Die Tatsache, dass das Einschalten der Spule den Kontakt öffnet, der sie speist, wird bis zum nächsten Durchgang ignoriert. Beim nächsten Durchgang sieht die SPS, dass der Kontakt geöffnet ist, und schaltet die Spule ab. Das Relais schaltet also immer noch schnell zwischen Ein und Aus um, allerdings in einem Rhythmus, der davon abhängt, wie oft die SPS den Stromkreis auswertet.

In HAL ist die Funktion der Code, der die Sprosse(n) auswertet. In der Tat exportiert die HAL-fähige Echtzeitversion von ClassicLadder eine Funktion, die genau das tut. In der Zwischenzeit ist ein Thread derjenige, der die Funktion in bestimmten Zeitintervallen ausführt. Genauso wie Sie wählen können, ob eine SPS alle 10&8239;ms oder jede Sekunde alle Sprossen auswerten soll, können Sie HAL-Threads mit unterschiedlichen Zeitabständen definieren.

Was einen Thread von einem anderen unterscheidet, ist "nicht" das, was der Thread tut - das wird dadurch bestimmt, welche Funktionen mit ihm verbunden sind. Der eigentliche Unterschied ist einfach, wie oft ein Thread läuft.

In LinuxCNC könnten Sie einen 50 µs Thread und einen 1 ms Thread haben. Diese würden basierend auf BASE_PERIOD und SERVO_PERIOD erstellt werden, die tatsächlichen Zeiten hängen von den Werten in Ihrer INI-Datei.

Der nächste Schritt ist zu entscheiden, was jeder Thread zu tun hat. Einige dieser Entscheidungen sind die gleichen in (fast) jeder LinuxCNC-System. Zum Beispiel wird motion-command-handler immer Servo-thread hinzugefügt.

Andere Verbindungen werden vom Integrator hergestellt. Dazu könnte gehören, dass die Encoder-Lese- und DAC-Schreibfunktionen des STG-Treibers mit dem Servo-Thread verbunden werden, oder dass die Stepgen-Funktion mit dem Base-Thread verbunden wird, zusammen mit der/den Parport-Funktion(en), um die Steps in den Port zu schreiben.

5.2. HAL Grundlagen

Dieses Dokument bietet einen Überblick über die Grundlagen von HAL.

5.2.1. HAL Befehle

Ausführlichere Informationen finden Sie in der Manpage für halcmd: führen Sie *man halcmd* in einem Terminalfenster aus.

Um die HAL-Konfiguration zu sehen und den Status von Pins und Parametern zu überprüfen, verwenden Sie das Fenster HAL-Konfiguration im Menü Maschine in AXIS. Um den Status eines Pins zu überwachen, öffnen Sie die Registerkarte "Überwachen" und klicken Sie auf jeden Pin, den Sie überwachen möchten; er wird dann zum Überwachungsfenster hinzugefügt.

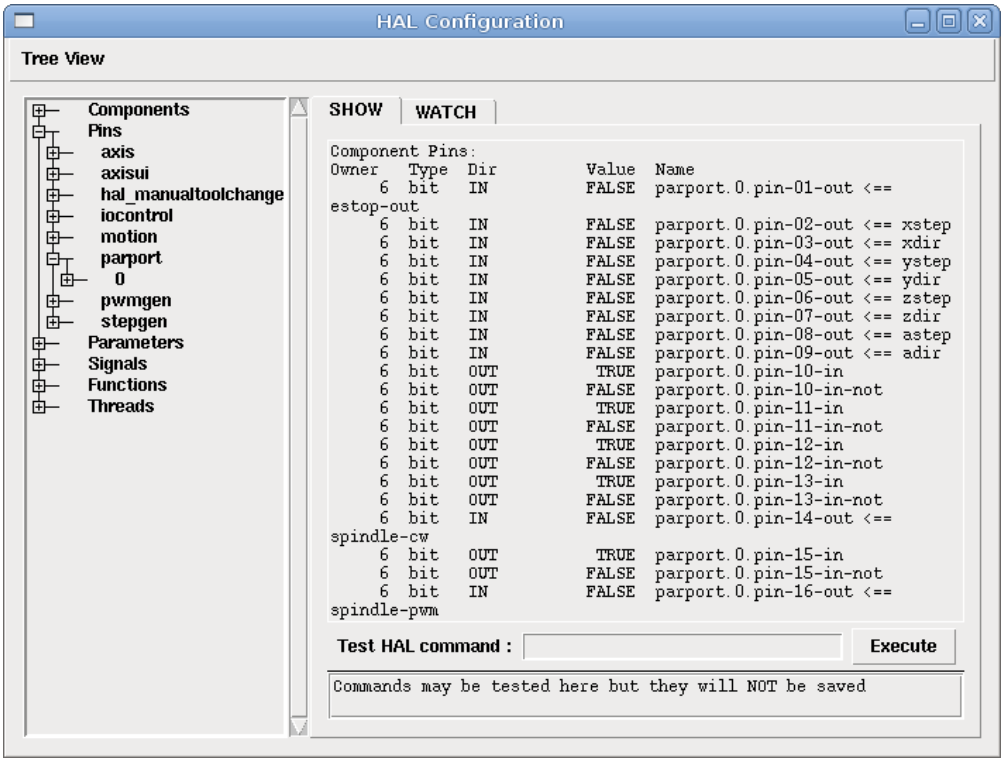


Figure 65. HAL-Konfigurationsfenster

loadrt

Mit dem Befehl `loadrt` wird eine Echtzeit-HAL-Komponente geladen. Echtzeit-Komponentenfunktionen müssen einem Thread hinzugefügt werden, um mit der Rate des Threads aktualisiert zu werden. Sie können keine Nicht-Echtzeit-Komponente in den Echtzeitbereich laden.

loadrt Syntax und Beispiel

```
loadrt <component> <options>
loadrt mux4 count=1
```

addf

Der Befehl `addf` fügt eine Funktion zu einem Echtzeit-Thread hinzu. Wenn der StepConf-Assistent zur Erstellung der Konfiguration verwendet wurde, wurden zwei Threads erstellt (``base-thread`` und ``servo-thread``).

`addf` fügt Funktion *Funktionsname* zu Thread *Threadname* hinzu. Standardmäßig wird die Funktion in der Reihenfolge hinzugefügt, in der sie in der Datei steht. Wenn *Position* angegeben ist, wird die Funktion an dieser Stelle des Threads hinzugefügt. Eine negative *Position* gibt die Position in Bezug auf das Ende des Threads an. Zum Beispiel *1* ist der Anfang des Threads, *-1* ist das Ende des Threads, *-3* ist das dritte Ende vom Ende.

Bei einigen Funktionen ist es wichtig, dass sie in einer bestimmten Reihenfolge geladen werden, wie z. B. die Parport-Lese- und Schreibfunktionen. Der Funktionsname ist normalerweise der Komponentenname plus eine Zahl. Im folgenden Beispiel wird die Komponente "or2" geladen und `show function` zeigt den Namen der or2-Funktion an.

```
$ halrun
halcmd: loadrt or2
halcmd: show function
Exported Functions:
Owner   CodeAddr  Arg       FP   Users  Name
 00004  f8bc5000  f8f950c8  YES      0   or2.0
```

You have to add a function from a HAL real time component to a thread to get the function to update at the rate of the thread. Usually there are two threads as shown in this example.

NOTE The `FP` column always shows *YES* and will be removed in the future. All threads unconditionally save and restore floating point state. The old `fp`/`nofp` distinction has been removed.

```
$ halrun
halcmd: loadrt motmod base_period_nsec=55555 servo_period_nsec=1000000 num_joints=3
halcmd: show thread
Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     995976  YES          servo-thread (        0,        0 )
      55332  YES           base-thread (        0,        0 )
```

- base-thread (the high-speed thread): This thread handles items that need a fast response, like making step pulses, and reading and writing the parallel port.
- servo-thread (the slow-speed thread): This thread handles items that can tolerate a slower response, like the motion controller, ClassicLadder, and the motion command handler.

addf Syntax und Beispiel

```
addf <function> <thread>
addf mux4.0 servo-thread
```

initf

The `initf` command registers a function to run once in realtime context, on a dedicated init cycle of the thread before the cyclic function list runs. It is the realtime-thread analogue of `addf`, intended for one-shot setup that must execute in the realtime task (for example EtherCAT master activation via `lcec.activate`).

`initf` adds function *functname* to the init list of thread *threadname*. The init list runs once on the first cycle after `start`, then is drained. The cyclic list begins on the following period. Once the init cycle has run, further `initf` calls on that thread are rejected. `position` has the same meaning as in `addf`.

initf Syntax and Example

```
initf <function> <thread>
initf lcec.activate servo-thread
```

loadusr

Der Befehl `loadusr` lädt eine Nicht-Echtzeit-HAL-Komponente. Nicht-Echtzeit-Programme sind ihre eigenen, separaten Prozesse, die optional mit anderen HAL-Komponenten über Pins und Parameter kommunizieren. Sie können keine Echtzeitkomponenten in den Nicht-Echtzeitbereich laden.

Flags können eine oder mehrere der folgenden sein:

-W	um auf die Bereitschaft der Komponente zu warten. Es wird davon ausgegangen, dass die Komponente denselben Namen hat wie das erste Argument des Befehls.
-Wn <Name>	um auf die Komponente zu warten, die den angegebenen <Name> haben wird. Dies gilt nur, wenn die Komponente eine Namensoption hat.
-w	um zu warten, bis das Programm beendet wird
-i	um den Rückgabewert des Programms zu ignorieren (mit -w)
-n	Benennt eine Komponente, sofern dies eine zulässige Option für diese Komponente ist.

Syntax und Beispiele für `loadusr`

```
loadusr <component> <options>
loadusr halui
loadusr -Wn spindle gs2_vfd -n spindle
```

Auf Deutsch bedeutet es *loadusr wartet auf Name Spindel Komponente gs2_vfd mit Namen Spindel.*

net

Der Befehl `net` erstellt eine *Verbindung* zwischen einem Signal und einem oder mehreren Pins. Wenn das Signal nicht existiert, erzeugt net das neue Signal. Dies ersetzt die Verwendung des Befehls newsig. Die optionalen Richtungspfeile `<=`, `=>` und `<=>` erleichtern das Verfolgen der Logik beim Lesen einer `net` -Befehlszeile und werden vom Befehl `net` nicht verwendet. Die Richtungspfeile müssen durch ein Leerzeichen von den Pin-Namen getrennt werden.

Syntax und Beispiele für `net`

```
net signal-name pin-name <optional arrow> <optional second pin-name>
net home-x joint.0.home-sw-in <= parport.0.pin-11-in
```

Im obigen Beispiel ist `home-x` der Signalname, `joint.0.home-sw-in` ist ein *Direction IN*-Pin, `<=` ist der optionale Richtungspfeil, und `parport.0.pin-11-in` ist ein *Direction OUT*-Pin. Dies mag verwirrend erscheinen, aber die Bezeichnungen "in" und "out" für einen Parallelport-Pin geben die physikalische

Funktionsweise des Pins an, nicht wie er in HAL gehandhabt wird.

Ein Pin kann mit einem Signal verbunden werden, wenn er die folgenden Regeln beachtet:

- Ein IN-Pin kann immer mit einem Signal verbunden werden.
- Ein IO-Pin kann angeschlossen werden, sofern kein ein OUT-Pin am Signal anliegt.
- Ein OUT-Pin kann nur angeschlossen werden, wenn es keine anderen OUT- oder IO-Pins am Signal gibt.

Derselbe *Signal-Name* kann in mehreren Netzbefehlen verwendet werden, um zusätzliche Pins zu verbinden, solange die obigen Regeln beachtet werden.

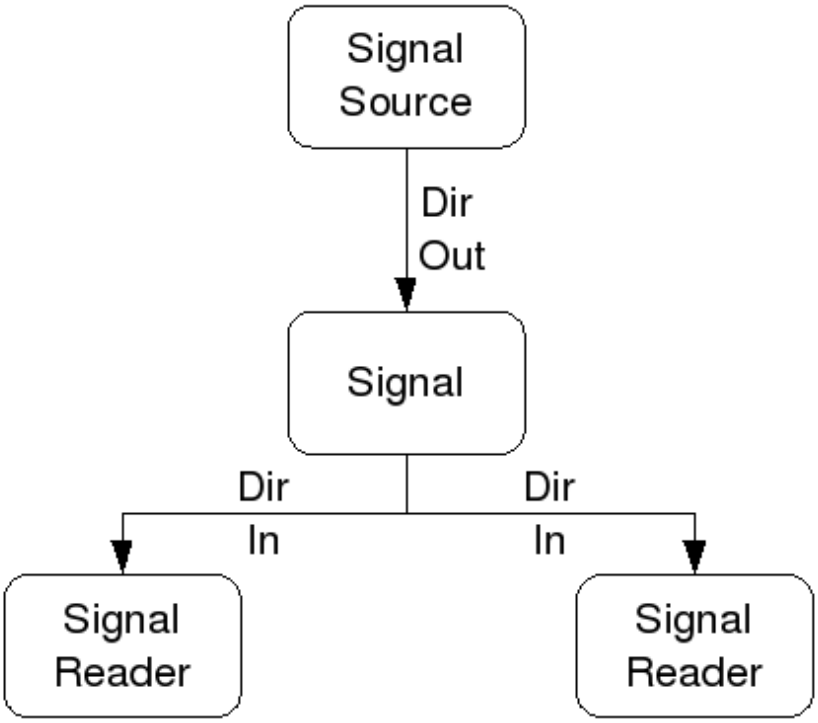


Figure 66. Signalrichtung (engl. signal direction)

Dieses Beispiel zeigt das Signal xStep mit der Quelle `stepgen.0.out` und mit zwei Lesern, `parport.0.pin-02-out` und `parport.0.pin-08-out`. Im Grunde genommen wird der Wert von `stepgen.0.out` an das Signal xStep gesendet und dieser Wert wird dann an `parport.0.pin-02-out` und `parport.0.pin-08-out gesendet`.

```
#   Signal    Ursprung           Destination          Destination
net xStep stepgen.0.out => parport.0.pin-02-out parport.0.pin-08-out
```

Da das Signal xStep den Wert von `stepgen.0.out` (die Quelle/Ursprung (engl. source)) enthält, können Sie dasselbe Signal erneut verwenden, um den Wert an einen anderen Leser zu senden. Verwenden Sie dazu einfach das Signal mit den Lesern in einer anderen Zeile.

```
#   Signal       Destination2
```

```
net xStep => parport.0.pin-06-out
```

E/A-Pins (engl. I/O Pins)

Ein E/A-Pin wie ein Encoder. *N*.index-enable kann gelesen oder so eingestellt werden, wie es die Komponente zulässt.

setp

Der Befehl `setp` setzt den Wert eines Pins oder Parameters. Die gültigen Werte hängen vom Typ des Pins oder Parameters ab. Es ist ein Fehler, wenn die Datentypen nicht übereinstimmen.

Einige Komponenten haben Parameter, die vor der Verwendung eingestellt werden müssen. Die Parameter können je nach Bedarf vor der Verwendung oder während der Ausführung gesetzt werden. Sie können setp nicht auf einen Pin anwenden, der mit einem Signal verbunden ist.

Syntax und Beispiele für `setp`

```
setp <pin/parameter-name> <value>
setp parport.0.pin-08-out TRUE
```

sets

Der Befehl `sets` setzt den Wert eines Signals.

Syntax und Beispiele für `sets`

```
sets <signal-name> <value>
net mysignal and2.0.in0 pyvcp.my-led
sets mysignal 1
```

Es ist ein Fehler, wenn:

- Der Signal-Name existiert nicht
- Wenn das Signal bereits einen Schreiber (engl. writer) hat
- Wenn Wert nicht der richtige Typ für das Signal ist

unlinkp

Der Befehl `unlinkp` löst die Verknüpfung eines Pins vom angeschlossenen Signal. Wenn vor dem Ausführen des Befehls kein Signal mit dem Pin verbunden war, passiert nichts. Der Befehl `unlinkp` ist nützlich für die Fehlerbehebung.

Syntax und Beispiele für `unlinkp`

```
unlinkp <pin-name>
unlinkp parport.0.pin-02-out
```

Veraltete Befehle

Die folgenden Befehle sind veraltet und werden möglicherweise aus zukünftigen Versionen entfernt. Jede neue Konfiguration sollte den Befehl `net` verwenden. Diese Befehle sind enthalten, damit ältere Konfigurationen noch funktionieren.

linksp (veraltet)

Der Befehl `linksp` stellt eine *Verbindung* (engl. connection) zwischen einem Signal und einem Pin her.

Syntax und Beispiele für `linksp`

```
linksp <signal-name> <pin-name>
linksp X-step parport.0.pin-02-out
```

Der Befehl `linksp` wurde durch den Befehl `net` abgelöst.

linkps (veraltet)

Der Befehl `linkps` stellt eine *Verbindung* zwischen einem Pin und einem Signal her. Er ist der gleiche wie `linksp`, aber die Argumente sind umgekehrt.

Syntax und Beispiele für `linkps`

```
linkps <pin-name> <signal-name>
linkps parport.0.pin-02-out X-Step
```

Der Befehl `linkps` wurde durch den Befehl `net` abgelöst.

newsig

the command `newsig` creates a new HAL signal by the name <*signame*> and the data type of <*type*>. Type must be *bool*, *sint*, *uint* or *real*. Error if <*signame*> already exists.

Syntax und Beispiele für `newsig`

```
newsig <signame> <type>
newsig Xstep bool
```

Weitere Informationen finden Sie im HAL-Handbuch oder in den Man Pages für `halrun`.

5.2.2. HAL Daten

bool

A bool value is an on or off.

- bool values = true or 1 and false or 0 (True, TRUE, true are all valid)

real

A *real* is a floating point number. In other words the decimal point can move as needed.

- real values = a double precision floating point value, with approximately 53 bits of resolution and over 2^{10} (~ 1000) bits of dynamic range.

Weitere Informationen über Gleitkommazahlen finden Sie unter:

https://en.wikipedia.org/wiki/Floating_point

sint

An *sint* number is a whole number that can have a negative or positive value.

- sint values = integer numbers from -9223372036854775808 to 9223372036854775807

uint

A *uint* number is a whole number that is positive only.

- uint values = integer numbers from 0 to 18446744073709551615

5.2.3. HAL Dateien

Wenn Sie den Stepper Config Wizard verwendet haben, um Ihre Konfiguration zu erstellen, werden Sie bis zu drei HAL-Dateien in Ihrem Konfigurationsverzeichnis haben.

- *my-mill.hal* (wenn Ihre Konfiguration *my-mill* heißt) Diese Datei wird zuerst geladen und sollte nicht geändert werden, wenn Sie den Stepper-Konfigurationsassistenten verwendet haben.
- *custom.hal* Diese Datei wird als nächstes und vor dem Laden der grafischen Benutzeroberfläche geladen. Hier legen Sie Ihre benutzerdefinierten HAL-Befehle ab, die vor dem Laden der grafischen Benutzeroberfläche geladen werden sollen.
- *custom_postgui.hal* Diese Datei wird geladen, nachdem die grafische Benutzeroberfläche geladen wurde. Hier werden die benutzerdefinierten HAL-Befehle abgelegt, die nach dem Laden der grafischen Benutzeroberfläche geladen werden sollen. Alle HAL-Befehle, die PyVCP-Widgets verwenden, müssen hier abgelegt werden.

5.2.4. Function Timing Pins

Three pins are automatically added to each HAL component when it is created. These allow you to scope the execution time of a component.

`.time`	Pin time is a *real* showing in seconds how long it took to execute the function.
`.tmax`	Pin tmax is a *real* holding the maximum time in seconds it took to execute the function.
`.tmax-increased`	This pin is set to true for one cycle if tmax increased.

`tmax` is an I/O pin, so the user can set it to 0 to get rid of the first time initialization on the function's execution time.

5.2.5. Grundlegende Logik-Komponenten

HAL enthält mehrere Echtzeit-Logikkomponenten. Logikkomponenten folgen einer "Wahrheitstabelle", die angibt, was die Ausgabe für eine bestimmte Eingabe ist. In der Regel handelt es sich dabei um Bitmanipulatoren, die elektrischen Logikgatter-Wahrheitstabellen folgen.

Für weitere Komponenten siehe HAL Components List oder die man pages.

and2

Die Komponente `and2` ist ein und-Gatter mit zwei Eingängen. Die folgende Wahrheitstabelle zeigt die Ausgabe für jede Kombination von Eingängen.

Syntax

```
and2 [count=N] | [names=name1[,name2...]]
```

Funktionen

```
and2.n
```

Pins

```
and2.N.in0 (bool, in)
and2.N.in1 (bool, in)
and2.N.out (bool, out)
```

Table 5. Wahrheitstabelle von `and2`

in0	in1	out
False	False	False
True	False	False
False	True	False
True	True	True

not

The `not` component is a boolean inverter.

Syntax

```
not [count=n] | [names=name1[,name2...]]
```

Funktionen

```
not.all
not.n
```

Pins

```
not.n.in (bool, in)
not.n.out (bool, out)
```

Table 6. Wahrheitstabelle von `not`

in	out
True	False
False	True

or2

Die `or2`-Komponente ist ein oder-Gatter mit zwei Eingängen.

Syntax

```
or2[count=n] | [names=name1[,name2...]]
```

Funktionen

```
or2.n
```

Pins

```
or2.n.in0 (bool, in)
or2.n.in1 (bool, in)
or2.n.out (bool, out)
```

Table 7. or2 Wahrheitstabelle

in0	in1	out
True	False	True
True	True	True
False	True	True
False	False	False

xor2

Die `xor2`-Komponente ist ein xor-Gatter (entweder oder) mit zwei Eingängen.

Syntax

```
xor2[count=n] | [names=name1[,name2...]]
```

Funktionen

```
xor2.n
```

Pins

```
xor2.n.in0 (bool, in)
xor2.n.in1 (bool, in)
xor2.n.out (bool, out)
```

Table 8. xor2-Wahrheitstabelle

in0	in1	out
True	False	True
True	True	False
False	True	True
False	False	False

5.2.6. Logik-Beispiele

.Beispiel mit `and2`

```
loadrt and2 count=1
addf and2.0 servo-thread
net my-sigin1 and2.0.in0 <= parport.0.pin-11-in
net my-sigin2 and2.0.in1 <= parport.0.pin-12-in
net both-on parport.0.pin-14-out <= and2.0.out
```

In the above example one copy of `and2` is loaded into real time space and added to the servo thread. Next `pin-11` of the parallel port is connected to the `in0` input of the and gate. Next `pin-12` is connected to the `in1` input of the and gate. Last we connect the `and2` output to the parallel port `pin-14`. So following the truth table for `and2` if pin 11 and pin 12 are on then the output pin 14 will be on.

5.2.7. Umwandlungs (engl. conversions)-Komponenten

weighted_sum

The weighted sum converts a group of booleans into an integer. The conversion is the sum of the *weights* of the booleans present plus any offset. It's similar to *binary coded decimal* but with more options. The *hold* input interrupts the input processing, so that the *sum* value no longer changes.

Syntax für das Laden der Komponente `weighted_sum`

```
loadrt weighted_sum wsum_sizes=size[,size,...]
```

Creates groups of ``weighted_sum``s, each with the given number of input booleans (size).

Um die "weighted_sum" zu aktualisieren, muss der "process_wsums" an einen Thread angehängt werden.

Hinzufügen von `process_wsums` *zum Servo-Thread*

```
addf process_wsums servo-thread
```

Which updates the `weighted_sum` component.

In the following example, a copy of the AXIS HAL configuration window, bools *0* and *2* are TRUE, they have no offset. The weight (*weight*) of bool 0 is 1, that of bool 2 is 4, so the sum is 5.

Table 9. Komponenten-Pins von `weighted_sum`

Eigentümer (engl. owner)	Typ	Richt	Wert	Name
10	bool	In	TRUE	wsum.0.bit.0.in
10	sint	E/A (engl. I/O)	1	wsum.0.bit.0.weight
10	bool	In	FALSE	wsum.0.bit.1.in
10	sint	E/A (engl. I/O)	2	wsum.0.bit.1.weight
10	bool	In	TRUE	wsum.0.bit.2.in
10	sint	E/A (engl. I/O)	4	wsum.0.bit.2.weight
10	bool	In	FALSE	wsum.0.bit.3.in
10	sint	E/A (engl. I/O)	8	wsum.0.bit.3.weight
10	bool	In	FALSE	wsum.0.hold
10	sint	E/A (engl. I/O)	0	wsum.0.offset
10	sint	Out	5	wsum.0.sum

5.3. HAL TWOPASS

5.3.1. TWOPASS

Dieser Abschnitt beschreibt eine Möglichkeit, mehrere Ladebefehle für mehrere Instanzen der gleichen Komponente an verschiedenen Positionen in der Datei oder zwischen verschiedenen Dateien zu haben. Intern erfordert dies, die HAL-Datei zweimal zu lesen, daher der Name TWOPASS. Unterstützt seit LinuxCNC Version 2.5, hilft die TWOPASS Verarbeitung von LinuxCNC Konfigurationsdateien durch eine leichtere Modularisierung und verbesserte Lesbarkeit. Zur Erinnerung, LinuxCNC Konfigurationsdateien sind in einem LinuxCNC INI-Datei als `[HAL]HALFILE=filename` angegeben.

Normalerweise muss ein Satz von einem oder mehreren LinuxCNC-Konfigurationsdateien eine einzige, eindeutige `loadrt`-Zeile verwenden, um eine Echtzeitkomponente zu laden, die mehrere Instanzen der Komponente erzeugen kann. Zum Beispiel, wenn Sie eine Zwei-Eingang UND-Gatter-Komponente (and2) in drei verschiedenen Orten in Ihrem Setup verwenden, müssten Sie eine einzige Zeile irgendwo zu spezifizieren:

Beispiel, das zu Echtzeitkomponenten mit den Standardnamen `and2.0`, `and2.1`, `and2.2` führt.

```
loadrt and2 count=3
```

Konfigurationen sind besser lesbar, wenn Sie die Option `names=` für Komponenten angeben, bei denen sie unterstützt wird, z. B.:

Beispiel für einen Ladebefehl, der zu explizit benannten Komponenten `aa`, `ab`, `ac` führt.

```
loadrt and2 names=aa,ab,ac
```

Es kann ein Wartungsproblem sein, den Überblick über die Komponenten und ihre Namen zu behalten, denn wenn Sie eine Komponente hinzufügen (oder entfernen), müssen Sie die einzelne loadrt-Anweisung für die Komponente finden und aktualisieren.

Die TWOPASS-Verarbeitung wird aktiviert, indem ein INI-Dateiparameter in den `[HAL]`-Abschnitt aufgenommen wird, wobei "anystring" eine beliebige Zeichenkette sein kann, die nicht null ist.

```
[HAL]

TWOPASS = anystring
```

Wenn TWOPASS aktiviert ist, können Sie mehrere Angaben machen:

```
loadrt and2 names=aa
...
loadrt and2 names=ab,ac
...
loadrt and2 names=ad
```

Diese Befehle können in verschiedenen HAL-Dateien vorkommen. Die HAL-Dateien werden in der Reihenfolge ihres Auftretens in der INI-Datei abgearbeitet, bei Mehrfachzuweisungen von HALFILE.

Die Option TWOPASS kann mit Optionen angegeben werden, um Ausgaben für die Fehlersuche hinzuzufügen (verbose) und um das Löschen temporärer Dateien zu verhindern (nodelete). Die Optionen werden durch Kommata getrennt.

Beispiel

```
[HAL]
TWOPASS = on,verbose,nodelete
```

Bei der TWOPASS-Verarbeitung werden zunächst alle [HAL]HALFILES gelesen und die mehrfachen Auftritte der loadrt-Anweisungen für jedes Modul kumuliert. Nicht-Echtzeit-Komponenten (loadusr) werden der Reihe nach geladen, aber keine anderen LinuxCNC-Befehle werden im ersten Durchgang ausgeführt.

NOTE	Nicht-Echtzeit-Komponenten sollten die Option wait (-W) verwenden, um sicherzustellen, dass die Komponente bereit ist, bevor andere Befehle ausgeführt werden.

Nach dem ersten Durchlauf werden die Echtzeitmodule automatisch geladen (loadrt)

- mit einer Zahl, die gleich der Gesamtzahl ist, wenn die Option »count=« verwendet wird, oder
- mit allen einzelnen Namen, die bei Verwendung der Option „names=“ angegeben werden.

In einem zweiten Durchlauf werden dann alle anderen in den HALFILES angegebenen LinuxCNC-Befehle ausgeführt. Die addf-Befehle verknüpfen die Funktionen einer Komponente mit der Thread-Ausführung und werden in diesem zweiten Durchgang in der Reihenfolge ihres Erscheinens zusammen mit anderen Befehlen ausgeführt.

Die Optionen "count=" und "names=" können zwar verwendet werden, schließen sich aber gegenseitig aus - für ein bestimmtes Modul kann nur ein Typ angegeben werden.

Die TWOPASS-Verarbeitung ist am effektivsten, wenn die Option "names=" verwendet wird. Mit dieser Option können Sie eindeutige Namen vergeben, die als Gedächtnisstütze dienen oder anderweitig für die Konfiguration relevant sind. Wenn Sie z. B. eine Ableitungskomponente zur Schätzung der Geschwindigkeiten und Beschleunigungen an jeder (x,y,z)-Koordinate verwenden, führt die Verwendung der *count*=-Methode zu obskuren Komponentennamen wie ddt.0, ddt.1, ddt.2, usw.

Alternativ können Sie auch die Option *names*= verwenden:

```
loadrt ddt names=xvel,yvel,zvel
...
loadrt ddt names=xaccel,yaccel,zaccel
```

führt zu Komponenten mit den sinnvollen Namen xvel, yvel, zvel, xaccel, yaccel, zaccel.

Viele Comps, die mit der Distribution geliefert werden, wurden mit dem halcompile Dienstprogramm erstellt und unterstützen die Option *names*=. Dazu gehören die gemeinsamen Logik-Komponenten, die der Klebstoff von vielen LinuxCNC-Konfigurationen sind.

Vom Benutzer erstellte Kompilate, die das Dienstprogramm halcompile verwenden, unterstützen automatisch auch die Option names=. Neben den mit dem Dienstprogramm halcompile erstellten Comps unterstützen auch zahlreiche andere Comps die Option names=. Zu den Comps, welche die Option names= unterstützen, gehören: at_pid, encoder, encoder_ratio, pid, siggen und sim_encoder.

Die Verarbeitung erfolgt in zwei Schritten, bevor das GUI geladen wird. Bei Verwendung einer [HAL]POSTGUI_HALFILE ist es zweckmäßig, alle [HAL]POSTGUI_HALFILE-Loadrt-Deklarationen für die erforderlichen Komponenten in einer vorgeladenen HAL-Datei unterzubringen.

Beispiel für einen HAL-Abschnitt bei Verwendung einer POSTGUI_HALFILE

```
[HAL]

TWOPASS = on
HALFILE = core_sim.hal
HALFILE = sim_spindle_encoder.hal
HALFILE = axis_manualtoolchange.hal
HALFILE = simulated_home.hal
HALFILE = load_for_postgui.hal <- loadrt-Zeilen für Komponenten in postgui.hal

POSTGUI_HALFILE = postgui.hal
HALUI = halui
```

5.3.2. Post GUI (lat. für nach dem GUI auszuführen)

Einige GUIs unterstützen HAL-Dateien, die verarbeitet werden, nachdem die GUI gestartet wurde, um LinuxCNC-Pins zu verbinden, die von der GUI erzeugt wurden. Wenn Sie eine Postgui-HAL-Datei mit TWOPASS-Verarbeitung verwenden, fügen Sie alle loadrt-Elemente für Komponenten, die durch Postgui-HAL-Dateien hinzugefügt wurden, in ein separates HAL-Dateien ein, das vor der GUI verarbeitet wird. Die `addf`-Befehle können ebenfalls in diese Datei aufgenommen werden.

Beispiel

```
[HAL]
TWOPASS = on
HALFILE = file_1.hal
# ...
HALFILE = file_n.hal
HALFILE = file_with_all_loads_for_postgui.hal
# ...
POSTGUI_HALFILE = the_postgui_file.hal
```

5.3.3. Ausschließen von HAL-Dateien

Die TWOPASS-Verarbeitung konvertiert *.hal*-Dateien in äquivalente *.tcl*-Dateien und verwendet haltcl, um loadrt- und addf-Befehle zu finden, um ihre Nutzung zu akkumulieren und zu konsolidieren. Loadrt-Parameter, die den einfachen Parametern `names=` (oder `count=`) entsprechen, wie sie der HAL Component Generator (`halcompile`) akzeptiert, werden erwartet. Komplexere Parameterelemente, die in spezialisierten LinuxCNC-Komponenten enthalten sind, werden möglicherweise nicht ordnungsgemäß behandelt.

Eine HAL-Datei kann von der TWOPASS-Verarbeitung ausgeschlossen werden, indem man eine magische Kommentarzeile an beliebiger Stelle in die HAL-Datei einfügt. Die magische Kommentarzeile muss beginnen mit der Zeichenfolge *#NOTWOPASS* beginnen. Die mit diesem magischen Kommentar bedachten Dateien werden von halcmd unter Verwendung der Optionen `-k` (bei Fehlschlag weitermachen) und `-v` (verbose) gelesen.

Diese Ausschlussbestimmung kann verwendet werden, um Probleme zu isolieren oder um spezielle LinuxCNC-Komponenten zu laden, die keine TWOPASS-Verarbeitung benötigen oder davon profitieren.

Normalerweise ist die loadrt-Reihenfolge von Echtzeit-Komponenten nicht kritisch, aber die loadrt-Reihenfolge für spezielle Komponenten kann erzwungen werden, indem man die entsprechenden loadrt-Direktiven in einer ausgeschlossenen Datei platziert.

NOTE Während die Reihenfolge der loadrt-Direktiven in der Regel unkritisch ist, so ist die Reihenfolge der addf-Direktiven oft sehr wichtig für den ordnungsgemäßen Betrieb von Servo-Regelkreis-Komponenten.

Beispiel einer ausgeschlossenen HAL-Datei

```
$ cat twopass_excluded.hal
# Der folgende magische Kommentar bewirkt, dass diese Datei
# von der twopass-Verarbeitung ausgeschlossen wird:
# NOTWOPASS

# Komponente mit komplexen Optionen debuggen:
loadrt mycomponent parm1="abc def" parm2=ghi
show pin mycomponent

# Ordnen spezieller Komponenten
loadrt Bauteil_1
loadrt Bauteil_2
```

NOTE Groß- und Kleinschreibung sowie Leerzeichen innerhalb des magischen Kommentars werden ignoriert. Das Laden von Komponenten, die `names=` oder `count=` Parameter verwenden (typischerweise von halcompile erstellt) sollte nicht in ausgeschlossenen Dateien verwendet werden, da dies die Vorteile der TWOPASS-Verarbeitung eliminieren würde. Die LinuxCNC-Befehle, die Signale erzeugen (`net`) und Befehle, welche die Ausführungsreihenfolge festlegen (`addf`), sollten nicht in ausgeschlossenen Dateien platziert werden. Dies gilt insbesondere für addf-Befehle, da ihre Reihenfolge wichtig sein kann.

5.3.4. Beispiele

Beispiele für die Verwendung von TWOPASS für einen Simulator sind in den Verzeichnissen enthalten:

```
configs/sim/axis/twopass/
configs/sim/axis/simtcl/
```

5.4. HAL Werkzeuge (engl. tools)

For most of the tools, a more detailed description can be found in the chapter HAL Tutorial

5.4.1. Halcmd

`halcmd` ist ein Kommandozeilen-Tool für die Manipulation des HAL. Es gibt eine ziemlich vollständige Manpage für `halcmd`, die installiert wird, wenn Sie LinuxCNC entweder aus dem Quellcode oder einem Paket installiert haben. Die Manpage bietet Informationen zur Benutzung:

```
man halcmd
```

Wenn Sie LinuxCNC für "run-in-place" kompiliert haben, müssen Sie das rip-environment Skript als Quelle angeben, um die Manpage verfügbar zu machen:

```
cd toplevel_directory_for_rip_build
. scripts/rip-environment
man halcmd
```

Das HAL Tutorial enthält eine Reihe von Beispielen für die Verwendung von halcmd und ist ein gutes Tutorial für `halcmd`.

5.4.2. Halmeter

`halmeter` is like a voltmeter for the HAL. It lets you look at a pin, signal, or parameter, and displays the current value of that item. It is pretty simple to use. Start it by typing `halmeter` in an X windows shell. Halmeter is a GUI application. It will pop up a small window, with two buttons labeled "Select" and "Exit". Exit is easy - it shuts down the program. Select pops up a larger window, with three tabs. One tab lists all the pins currently defined in the HAL. The next lists all the signals, and the last tab lists all the parameters. Click on a tab, then click on a pin/signal/parameter. Then click on "OK". The lists will disappear, and the small window will display the name and value of the selected item. The display is updated approximately 10 times per second. If you click "Accept" instead of "OK", the small window will display the name and value of the selected item, but the large window will remain on the screen. This is convenient if you want to look at a number of different items quickly.

Sie können viele Halbmesser gleichzeitig laufen lassen, wenn Sie mehrere Elemente überwachen möchten. Wenn Sie ein Halbmesser starten möchten, ohne ein Shell-Fenster zu kleben, geben Sie `halmeter & `, um es im Hintergrund auszuführen. Sie können auch die Halmeter-Startanzeige sofort vornehmen, indem Sie `pin|sig|par[am] _<name>_` in die Befehlszeile einfügen. Es wird den Pin, das Signal oder den Parameter <*name*> angezeigt, sobald er beginnt - wenn es keinen solchen Artikel gibt, wird es einfach normal starten. Und schließlich, wenn Sie ein Element zum Anzeigen angeben, können Sie `-s` vor dem `pinigsigparaparam` hinzufügen, um Halmeter zu sagen, um ein kleines Fenster zu verwenden. Der Artikelname wird in der Titelleiste anstatt unter dem Wert angezeigt, und es wird keine Schaltflächen. Nützlich, wenn Sie eine Menge von Metern in einer kleinen Menge Bildschirm Raum wollen.

In the Halmeter Tutorial you will find for more information.

`halmeter` kann von einem Terminal oder von AXIS geladen werden. `halmeter` ist bei der Anzeige von Werten schneller als `halshow`. `halmeter` hat zwei Fenster, eines zur Auswahl des zu überwachenden Pins, Signals oder Parameters und eines zur Anzeige des Wertes. Mehrere `halmeter` können gleichzeitig geöffnet sein. Wenn man ein Skript benutzt, um mehrere `halmeter` zu öffnen, kann man die Position jedes einzelnen mit `-g X Y` relativ zur oberen linken Ecke des Bildschirms festlegen. Zum Beispiel:

```
loadusr halmeter pin hm2.0.stepgen.00.velocity-fb -g 0 500
```

Siehe die Manpage für weitere Optionen und den Abschnitt Halmeter.

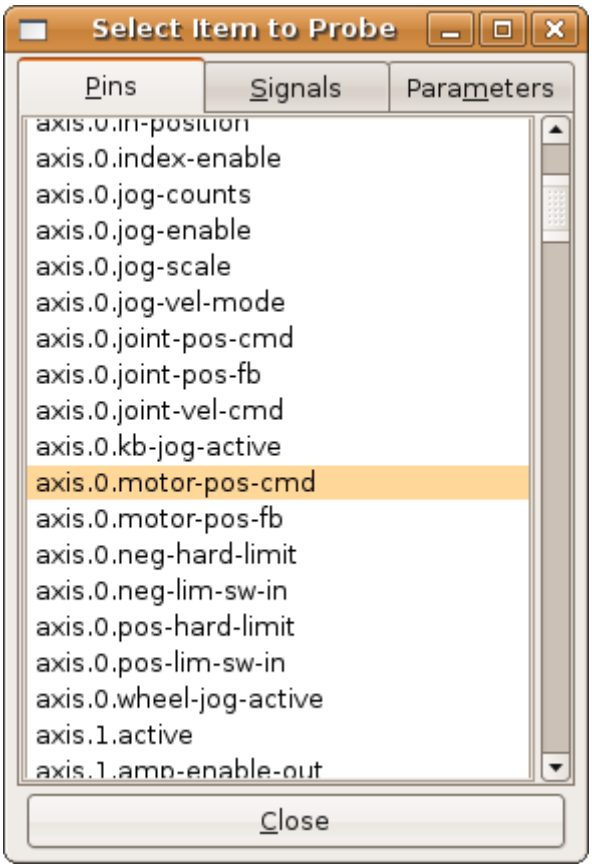


Figure 67. Halmeter-Auswahlfenster

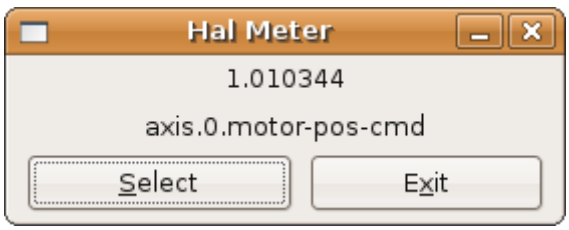


Figure 68. Halmeter-Watch Fenster

5.4.3. Halshow

`halshow` shows the values of chosen pins, parameters or signals of a running HAL. It further provides buttons to also modify those items. The WATCH tab provides a continuous display of selected pin, parameters, and signal items. The File menu provides buttons to save the watch items to a watch list and

to load an existing watch list. The watch list items can also be loaded automatically on startup.

More detailed information can be found in the section Halshow in the tutorial chapter.

It can be started from the command line:

```
halshow --help
Usage:
  halshow [Options] [watchfile]
  Options:
           --help    (this help)
           --fformat format_string_for_float
           --iformat format_string_for_int

Hinweise:
       Erstellen Sie einen watchfile in halshow mit: 'File/Save Watch List'.
       LinuxCNC muss für die Standalone-Nutzung ausgeführt werden.
```

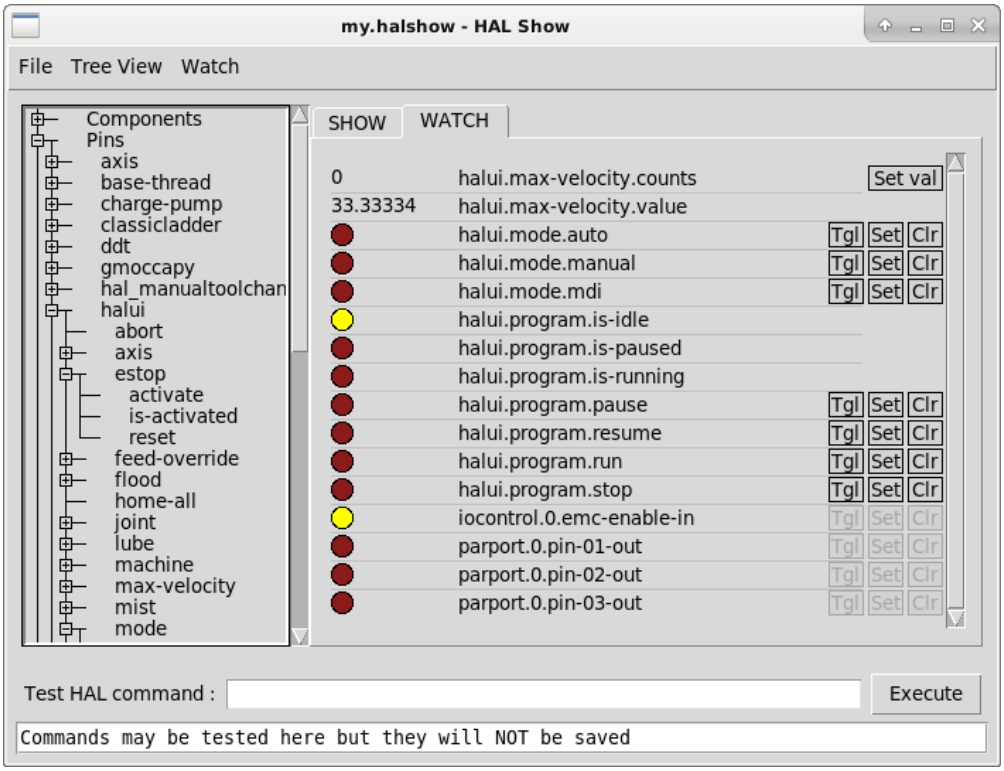


Figure 69. Halshow Watch Tab

Eine mit dem Menüpunkt *File/Save Watch List* erstellte Watchdatei wird als einzelne Zeile mit den Token "pin+", "param+", "sig=+", gefolgt vom entsprechenden Pin-, Param- oder Signalnamen formatiert. Die Token-Namen-Paare werden durch ein Leerzeichen getrennt.

Einzeiliges Watchfile-Beispiel

```
pin+joint.0.pos-hard-limit pin+joint.1.pos-hard-limit sig+estop-loop
```

Eine Überwachungsdatei, die mit dem Menüpunkt *Datei/Beobachtungsliste speichern (mehrzeilig)* erstellt wurde, wird mit separaten Zeilen für jedes Element formatiert, das wie oben beschrieben mit Token-Name-Paaren identifiziert wird.

Beispiel für eine Watchfile mit getrennten Zeilen

```
pin+joint.0.pos-hard-limit
pin+joint.1.pos-hard-limit
sig+estop-loop
```

Beim Laden einer Watchfile mit dem Menüpunkt *File/Load Watch List* können die Token-Namen-Paare als einzelne oder mehrere Zeilen erscheinen. Leerzeilen und Zeilen, die mit einem #-Zeichen beginnen, werden ignoriert.

5.4.4. Halscope

`halscope` ist ein *Oszilloskop* für den HAL. Damit können Sie den Wert von Pins, Signalen und Parametern als Funktion der Zeit aufzeichnen. Eine vollständige Bedienungsanleitung sollte hier zu finden sein. Für den Moment verweisen wir auf den Abschnitt Halscope im Kapitel Tutorial, in dem die Grundlagen erklärt werden.

Das halscope "Datei"-Menü bietet Schaltflächen zum Speichern einer Konfiguration oder zum Öffnen einer zuvor gespeicherten Konfiguration. Nach dem Beenden von halscope wird die letzte Konfiguration in einer Datei namens autosave.halscope gespeichert.

Konfigurationsdateien können auch beim Start von halscope über die Kommandozeile angegeben werden. Verwendung der Kommandozeilenhilfe (`-h`):

```
halscope -h
Usage:
  halscope [-h] [-i infile] [-o outfile] [num_samples]
```

5.4.5. Sim-Pin

sim_pin ist ein Kommandozeilenprogramm zur Anzeige und Aktualisierung einer beliebigen Anzahl von beschreibbaren Pins, Parametern oder Signalen.

sim_pin Verwendung

```
Usage:
        sim_pin [Options] name1 [name2 ...] &

Options:
        --help                (this text)
        --title title_string  (window title, default: sim_pin)

Hinweis: LinuxCNC (oder eine eigenständige HAL-Anwendung) muss ausgeführt werden
        Ein benanntes Element kann einen Pin, einen Parameter oder ein Signal angeben.
        Das Element muss beschreibbar sein, z. B:
          Pin:    IN oder I/O (und nicht mit einem Signal mit einem Schreiber verbunden)
          param:  RW
          Signal: verbunden mit einem beschreibbaren Pin

        HAL item types bool,sint,uint,real are supported.
```

```
        When a bool item is specified, a pushbutton is created
        to manage the item in one of three manners specified
        by radio buttons:
            toggle: Toggle value when button pressed
            pulse:  Pulse item to 1 once when button pressed
            hold:   Set to 1 while button pressed
        The bool pushbutton mode can be specified on the command
        line by formatting the item name:
            namei/mode=[toggle | pulse | hold]
        If the mode begins with an uppercase letter, the radio
        buttons for selecting other modes are not shown
```

Vollständige Informationen finden Sie in der Manpage:

```
man sim_pin
```

sim_pin Beispiel (mit laufendem LinuxCNC)

```
halcmd loadrt mux2 names=example; halcmd net sig_example example.in0
sim_pin example.sel example.in1 sig_example &
```

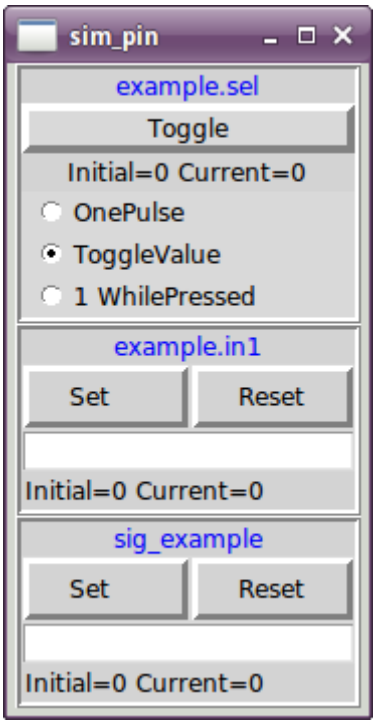


Figure 70. sim_pin Fenster

5.4.6. simulate_probe (Sonde simulieren)

`simulate_probe` ist ein einfaches GUI, um die Aktivierung des Pins motion.probe-input zu simulieren. Verwendung:

```
simulate_probe &
```

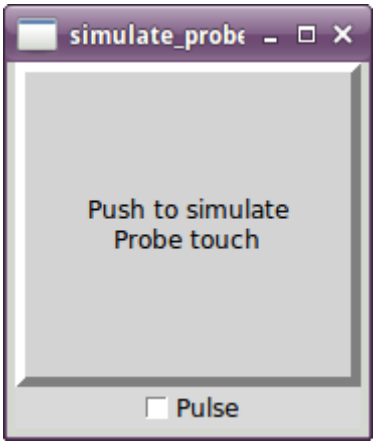


Figure 71. simulieren_probe Fenster

5.4.7. HAL Histogramm

`hal-histogram` ist ein Kommandozeilenprogramm zur Anzeige von Histogrammen für HAL-Pins.

Nutzung:

```
   hal-histogram --help | -?
or
   hal-histogram [Optionen] [Pinname]
```

Table 10. Optionen:

Option	Wert	Beschreibung
--minvalue	minvalue	minimum bin, Voreinstellung: 0
--binsize	binsize	binsize, default: 1e-07 (suits the default pin, a runtime in seconds)
--nbins	nbins	Anzahl der Bins, Voreinstellung: 50
--logscale	0/1	y-Achse logarithmische Skala, Voreintellung: 1
--text	Hinweis	Textanzeige, Voreinstellung: ""
-show		Anzahl der nicht angezeigten nbins anzeigen, Voreinstellung aus
--verbose		Fortschritt und Fehlersuche, standardmäßig aus

Anmerkungen:

1. LinuxCNC (oder eine andere HAL-Anwendung) muss laufen.
2. Wenn kein Pinname angegeben wird, lautet die Vorgabe: `motion-command-handler.time`.

3. Diese App kann für 5 Pins geöffnet werden.
4. Pintypes real, sint, uint, bool are supported.
5. The pin must be associated with a realtime thread.

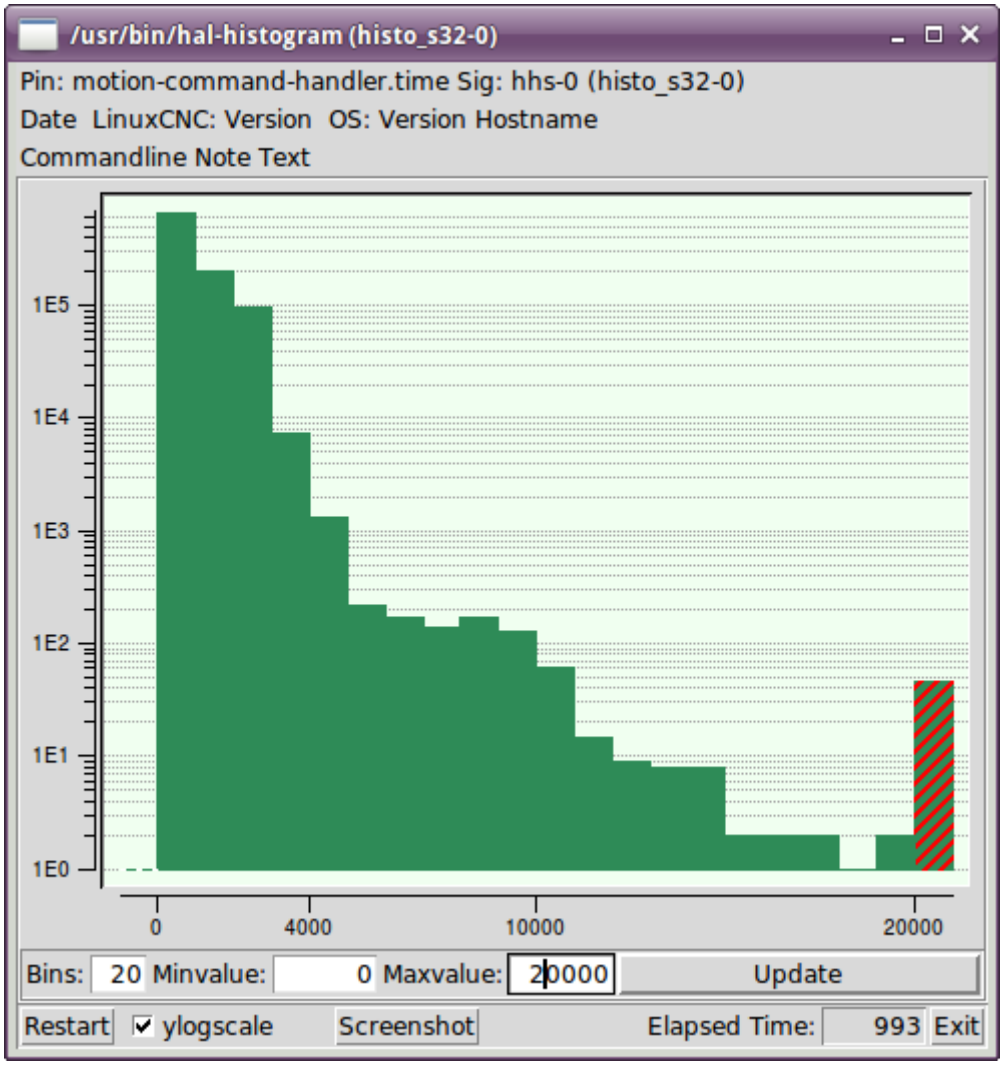


Figure 72. hal-histogram Fenster

5.4.8. Halreport

`halreport` ist ein Befehlszeilen-Dienstprogramm, das einen Bericht über HAL-Verbindungen für eine laufende LinuxCNC (oder andere HAL) Anwendung erzeugt. Der Bericht zeigt alle Signalverbindungen an und weist auf mögliche Probleme hin. Enthaltene Informationen:

1. Systembeschreibung und Kernelversion.
2. Signale und alle angeschlossenen Ausgangs-, E/A- und Eingangspins.
3. Eines jeden Pin component_function, thread und addf-Reihenfolge.
4. Nicht-Echtzeit-Komponentenpins mit nicht geordneten Funktionen.
5. Identifizierung von unbekannten Funktionen für nicht behandelte Komponenten.
6. Signale ohne Ausgang.
7. Signale ohne Eingänge.

8. Funktionen ohne addf.
9. Warnhinweise für Komponenten, die in der Dokumentation als veraltet/überflüssig gekennzeichnet sind.
10. Echte Namen für Pins, die Aliasnamen verwenden.

Der Bericht kann über die Befehlszeile erstellt und in eine Ausgabedatei (oder stdout, wenn kein Ausgabedateiname angegeben ist) geleitet werden:

halreport Anwendung

```
Usage:
  halreport -h | --help (this help)
or
  halreport [outfilename]
```

Um den Bericht für jeden LinuxCNC-Start zu erzeugen, fügen Sie halreport und einen Ausgabedateinamen als [APPLICATIONS]APP-Eintrag in die INI-Datei ein.

halreport Beispiel

```
[APPLICATIONS]
APP = halreport /tmp/halreport.txt
```

Die Reihenfolge, in der die Funktionen addiert werden, kann für Servoschleifen wichtig sein, bei denen die Reihenfolge der in jeder Servoperiode berechneten Funktionen wichtig ist. Typischerweise ist die Reihenfolge:

1. Eingangspins lesen,
2. die Funktionen für die Bewegungssteuerung und die Bewegungssteuerung ausführen,
3. PID-Berechnungen durchführen und schließlich
4. Ausgabepins schreiben.

Für jedes Signal in einem kritischen Pfad sollte die addf-Ordnung des Ausgangspins numerisch niedriger sein als die addf-Ordnung der kritischen Eingangspins, mit denen es verbunden ist.

Bei Routine-Signalpfaden, die Schalteingänge, Nicht-Echtzeit-Pins usw. verarbeiten, ist die addf-Reihenfolge oft nicht kritisch. Außerdem kann das Timing der Wertänderungen von Nicht-Echtzeit-Pins nicht in den für HAL-Threads typischen Intervallen kontrolliert oder garantiert werden.

Beispiel für eine PID-Schleife für einen hostmot2-Stepgen, der im Geschwindigkeitsmodus auf einer Trivkins-Maschine betrieben wird, wobei `joint.0` der X-Achsenkoordinate entspricht:

```
SIG:    pos-fb-0
  OUT:    h.00.position-fb                    hm2_7i92.0.read        servo-thread 001
          (=hm2_7i92.0.stepgen.00.position-fb)
    IN:     X_pid.feedback                    X_pid.do-pid-calcs     servo-thread 004
    IN:     joint.0.motor-pos-fb              motion-command-handler servo-thread 002
            ....................              motion-controller      servo-thread 003
...
```

```
SIG:    pos-cmd-0
  OUT:    joint.0.motor-pos-cmd               motion-command-handler servo-thread 002
          .....................               motion-controller      servo-thread 003
    IN:     X_pid.command                     X_pid.do-pid-calcs     servo-thread 004
...
SIG:    motor-cmd-0
  OUT:    X_pid.output                        X_pid.do-pid-calcs     servo-thread 004
    IN:     h.00.velocity-cmd                 hm2_7i92.0.write       servo-thread 008
            (=hm2_7i92.0.stepgen.00.velocity-cmd)
```

Im obigen Beispiel verwendet die HALFILE halcmd-Aliase, um die Pin-Namen für ein hostmot2-FPGA-Board mit Befehlen wie diesen zu vereinfachen:

```
alias pin hm2_7i92.0.stepgen.00.position-fb h.00.position-fb
```

NOTE

Eine fragwürdige Erkennung von Komponentenfunktionen kann auftreten bei

1. nicht unterstützte (veraltete) Komponenten,
2. vom Benutzer erstellte Komponenten, die mehrere Funktionen oder unkonventionelle Funktionsbezeichnungen verwenden, oder
3. GUI-erstellte nicht-Echtzeit-Komponenten, denen Unterscheidungsmerkmale wie ein Präfix auf Basis des GUI-Programmnamens fehlen.

Fragwürdige Funktionen sind mit einem Fragezeichen "?" gekennzeichnet.

NOTE

Komponentenstifte, die nicht mit einer bekannten Gewindefunktion verbunden werden können, melden die Funktion als "Unbekannt".

`halreport` generiert einen Verbindungsbericht (ohne Pintypen und aktuelle Werte) für eine laufende HAL-Anwendung, um den Entwurf und die Überprüfung von Verbindungen zu erleichtern. Dies hilft zu verstehen, was die Quelle eines Pin-Wertes ist. Verwenden Sie diese Informationen mit Anwendungen wie `halshow`, `halmeter`, `halscope` oder dem Befehl `halcmd show` in einem Terminal.

5.5. HAL Tutorial

5.5.1. Einführung

Die Konfiguration geht von der Theorie zum Gerät über - dem HAL-Gerät. Für diejenigen, die nur ein wenig Erfahrung mit Computerprogrammierung haben, ist dieser Abschnitt das "Hello World" des HAL.

`halrun` kann verwendet werden, um ein funktionierendes System zu erstellen. Es ist ein Kommandozeilen- oder Textdateiwerkzeug für Konfiguration und Tuning.

5.5.2. Halcmd

`halcmd` ist ein Befehlszeilentool zum Manipulieren von HAL. Eine vollständigere Manpage existiert für halcmd und wird zusammen mit LinuxCNC installiert, aus dem Quellcode oder aus einem Paket. Wenn LinuxCNC als *run-in-place* kompiliert wurde, wird die Manpage nicht installiert, ist aber im LinuxCNC-Hauptverzeichnis mit dem folgenden Befehl zugänglich:

```
$ man -M docs/build/man halcmd
```

Notation

In dieser Einführung werden die Befehle für das Betriebssystem in der Regel ohne die Eingabeaufforderung der UNIX-Shell gezeigt, d.h. typischerweise mit einem Dollarzeichen ($) oder einer Raute/Doppelkreuz (#). Bei der direkten Kommunikation mit dem HAL über `halcmd` oder `halrun` werden die Eingabeaufforderungen in den Beispielen gezeigt. Das Terminal-Fenster befindet sich unter "Anwendungen/Zubehör" in der Ubuntu-Menüleiste.

Terminalbefehl Beispiel - Eingabeaufforderungen

```
me@computer:~linuxcnc$ halrun
(wird wie die folgende Zeile angezeigt)
halrun

(die halcmd: Eingabeaufforderung wird beim Ausführen von HAL angezeigt)
halcmd: loadrt Zähler
halcmd: pin anzeigen
```

Befehl-Vervollständigung durch Tabulator-Taste

Ihre Version von `halcmd` enthält möglicherweise die Tabulator-Vervollständigung. Anstatt Dateinamen zu vervollständigen, wie es eine Shell tut, werden Befehle mit HAL-Kennungen vervollständigt. Sie müssen genügend Buchstaben eingeben, um eine eindeutige Übereinstimmung zu erzielen. Versuchen Sie, nach dem Start eines HAL-Befehls die Tabulatortaste zu drücken:

Befehl-Vervollständigung durch Tabulator-Taste

```
halcmd: loa<TAB>
halcmd: load
halcmd: loadrt
halcmd: loadrt cou<TAB>
halcmd: loadrt counter
```

Die RTAPI-Umgebung

RTAPI steht für Real Time Application Programming Interface. Viele HAL-Komponenten arbeiten in Echtzeit, und alle HAL-Komponenten speichern Daten im gemeinsamen Speicher, damit Echtzeitkomponenten darauf zugreifen können. Das normale Linux unterstützt keine Echtzeitprogrammierung oder die Art von gemeinsamem Speicher, die HAL benötigt. Glücklicherweise gibt es Echtzeitbetriebssysteme (RTOS) mit den notwendigen Erweiterungen für Linux. Leider geht jedes

RTOS die Dinge ein wenig anders an.

To address these differences, the LinuxCNC team came up with RTAPI, which provides a consistent way for programs to talk to the RTOS. If you are a programmer who wants to work on the internals of LinuxCNC, you may want to study *linuxcnc/include/rtapi.h* to understand the API. But if you are a normal person, all you need to know about RTAPI is that it (and the RTOS) needs to be loaded into the memory of your computer before you do anything with HAL.

5.5.3. Ein einfaches Beispiel

Laden einer Komponente

Für dieses Tutorial gehen wir davon aus, dass Sie die Live-CD erfolgreich installiert haben und, falls Sie eine RIP footnote: [Run In Place, wenn die Quelldateien in ein Benutzerverzeichnis heruntergeladen wurden und direkt von dort aus kompiliert und ausgeführt werden] Installation verwenden, das Skript "rip-environment" aufrufen, um Ihre Shell vorzubereiten. In diesem Fall müssen Sie nur noch die erforderlichen RTOS- und RTAPI-Module in den Speicher laden. Führen Sie einfach den folgenden Befehl in einem Terminalfenster aus:

HAL laden

```
cd linuxcnc
halrun
halcmd:
```

Nachdem das Echtzeitbetriebssystem und die RTAPI geladen sind, können wir mit dem ersten Beispiel beginnen. Beachten Sie, dass die Eingabeaufforderung jetzt als "halcmd:" angezeigt wird. Das liegt daran, dass die nachfolgenden Befehle als HAL-Befehle und nicht als Shell-Befehle interpretiert werden.

Für das erste Beispiel werden wir eine HAL-Komponente namens *siggen* verwenden, die ein einfacher Signalgenerator ist. Eine vollständige Beschreibung der Komponente *siggen* finden Sie im Abschnitt SigGen dieses Handbuchs. Es handelt sich um eine Echtzeit-Komponente. Um die Komponente "`siggen`" zu laden, verwenden Sie den HAL-Befehl `loadrt`.

Laden von siggen

```
halcmd: loadrt siggen
```

Untersuchung der HAL

Nun, nach dem Laden des Moduls, ist es an der Zeit, `halcmd` vorzustellen, das Kommandozeilenwerkzeug, das zur Konfiguration der HAL verwendet wird. Dieses Tutorial wird nur eine Auswahl der Funktionen von halcmd vorstellen. Eine ausführlichere Beschreibung finden Sie unter `man halcmd` oder in der Referenz im Abschnitt HAL Befehle (engl. commands) dieses Dokuments. Die erste halcmd-Funktion ist der *show*-Befehl. Dieser Befehl zeigt Informationen über den aktuellen Zustand der HAL an. Um alle installierten Komponenten anzuzeigen:

Komponenten mit `halrun`/`halcmd` *anzeigen*

```
halcmd: show comp

    Loaded HAL Components:
    ID      Type  Name                                PID   State
    3       RT    siggen                                    ready
    2       User  halcmd2177                          2177  ready
```

Da *halcmd* selbst auch eine HAL-Komponente ist, wird sie immer in der Liste erscheinen. Die Zahl hinter "`halcmd" in der Komponentenliste ist die UNIX-Prozess-ID. Da es möglich ist, mehr als eine Kopie von halcmd gleichzeitig laufen zu lassen (z.B. in verschiedenen Terminalfenstern), wird die PID an das Ende des Namens angehängt, um ihn eindeutig zu machen. Die Liste zeigt auch die Komponente "siggen", die wir im vorherigen Schritt installiert haben. Das "RT" unter "Typ" zeigt an, dass "siggen" eine Echtzeitkomponente ist. Das "User" unter "Type" zeigt an, dass es sich um eine nicht-Echtzeit-Komponente handelt.

Als Nächstes wollen wir sehen, welche Pins `siggen` zur Verfügung stellt:

Pins anzeigen

```
halcmd: show pin

Component Pins:
Owner   Type   Dir         Value  Name
     3  real   IN              1  siggen.0.amplitude
     3  bool   OUT         FALSE  siggen.0.clock
     3  real   OUT             0  siggen.0.cosine
     3  real   IN              1  siggen.0.frequency
     3  real   IN              0  siggen.0.offset
     3  real   OUT             0  siggen.0.sawtooth
     3  real   OUT             0  siggen.0.sine
     3  real   OUT             0  siggen.0.square
     3  real   OUT             0  siggen.0.triangle
     3  real   OUT             0  siggen.0.update.time
     3  real   IO              0  siggen.0.update.tmax
     3  bool   OUT         FALSE  siggen.0.update.tmax-increased
```

This command displays all of the pins in the current HAL. A complex system could have dozens or hundreds of pins. But right now there are only twelve pins. Of these pins ten are floating point and two are bool (boolean). Eight carry data out of the *siggen* component, three are used to transfer settings into the component, and one (*update.tmax*) is bidirectional. Since we have not yet executed the code contained within the component, some the pins have a value of zero.

Der nächste Schritt ist die Betrachtung der Parameter:

Parameter anzeigen

```
halcmd: show param

Parameters:
Owner   Type  Dir         Value   Name
```

The *show param* command shows all the parameters in the HAL. The *siggen* component has no parameters of its own, and the timing objects that come with a function are all pins, so the list is empty here. We will see some real parameters later, when *stepgen* is loaded. Note the column labeled *Dir*. Parameters labeled *RO* are read only parameters. They can be changed only by the component. Parameters labeled *RW* are read-write parameters. That means that they are changed by the component, but can also be changed by the user. Note: The pins `siggen.0.update.time`, `siggen.0.update.tmax` and `siggen.0.update.tmax-increased` are for debugging purposes and won't be covered in this section.

Die meisten Echtzeitkomponenten exportieren eine oder mehrere Funktionen, um den in ihnen enthaltenen Echtzeitcode tatsächlich auszuführen. Schauen wir uns an, welche Funktion(en) *siggen* exportiert hat:

Funktionen anzeigen mit `halcmd`

```
halcmd: show funct

Exported Functions:
Owner   CodeAddr  Arg       FP   Users  Name
00003   f801b000  fae820b8  YES      0  siggen.0.update
```

The siggen component exported a single function. It is not currently linked to any threads, so *users* is zero [1].

Echtzeitcode zum Laufen bringen

Um den in der Funktion `siggen.0.update` enthaltenen Code tatsächlich auszuführen, benötigen wir einen Echtzeit-Thread. Die Komponente namens *threads* wird zum Erstellen eines neuen Threads verwendet. Erstellen wir einen Thread namens "test-thread" mit einer Periode von 1 ms (1.000 µs oder 1.000.000 ns):

```
halcmd: loadrt threads name1=test-thread period1=1000000
```

Mal sehen, ob das funktioniert:

Threads anzeigen

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     999855  YES    test-thread        (        0,        0 )
```

It did. The period is not exactly 1,000,000 ns because of hardware limitations, but we have a thread that runs at approximately the correct rate. The next step is to connect the function to the thread:

Funktion hinzufügen

```
halcmd: addf siggen.0.update test-thread
```

Bis jetzt haben wir `halcmd` nur benutzt, um die HAL zu betrachten. Dieses Mal haben wir jedoch den Befehl `addf` (add function) verwendet, um tatsächlich etwas in der HAL zu ändern. Wir haben *halcmd* angewiesen, die Funktion `siggen.0.update` zum Thread *test-thread* hinzuzufügen, und wenn wir uns die Thread-Liste noch einmal ansehen, sehen wir, dass dies gelungen ist:

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     999855  YES    test-thread        (        0,        0 )
                  1 siggen.0.update
```

Bevor die Komponente *siggen* mit der Erzeugung von Signalen beginnt, ist noch ein weiterer Schritt erforderlich. Wenn die HAL zum ersten Mal gestartet wird, laufen die Threads noch nicht. Dies soll Ihnen ermöglichen, das System vollständig zu konfigurieren, bevor der Echtzeitcode startet. Sobald Sie mit der Konfiguration zufrieden sind, können Sie den Echtzeitcode wie folgt starten:

```
halcmd: start
```

Jetzt läuft der Signalgenerator. Schauen wir uns seine Ausgangspins an:

```
halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     3  real  IN              1  siggen.0.amplitude
     3  bool  OUT         FALSE  siggen.0.clock
     3  real  OUT    -0.1640929  siggen.0.cosine
     3  real  IN              1  siggen.0.frequency
     3  real  IN              0  siggen.0.offset
     3  real  OUT    -0.4475303  siggen.0.sawtooth
     3  real  OUT     0.9864449  siggen.0.sine
     3  real  OUT            -1  siggen.0.square
     3  real  OUT    -0.1049393  siggen.0.triangle
```

Und schauen wir noch einmal hin:

```
halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     3  real  IN              1  siggen.0.amplitude
     3  bool  OUT         FALSE  siggen.0.clock
     3  real  OUT     0.0507619  siggen.0.cosine
     3  real  IN              1  siggen.0.frequency
     3  real  IN              0  siggen.0.offset
     3  real  OUT     -0.516165  siggen.0.sawtooth
     3  real  OUT     0.9987108  siggen.0.sine
     3  real  OUT            -1  siggen.0.square
     3  real  OUT    0.03232994  siggen.0.triangle
```

Wir haben zwei `show pin`-Befehle kurz hintereinander ausgeführt, und Sie können sehen, dass die Ausgänge nicht mehr Null sind. Die Ausgänge für Sinus, Kosinus, Sägezahn und Dreieck ändern sich ständig. Der quadratische Ausgang funktioniert auch, aber er wechselt einfach bei jedem Zyklus von +1,0 auf -1,0.

Ändern von Parametern

Die eigentliche Stärke von HAL ist, dass man Dinge ändern kann. Wir können zum Beispiel den Befehl `setp` verwenden, um den Wert eines Parameters einzustellen. Ändern wir die Amplitude des Signalgenerators von 1,0 auf 5,0:

Pin einstellen (engl. set)

```
halcmd: setp siggen.0.amplitude 5
```

Überprüfen Sie die Parameter und Pins erneut

```
halcmd: show param

Parameters:
Owner   Type  Dir         Value  Name

halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     3  real  IN              5  siggen.0.amplitude
     3  bool  OUT         FALSE  siggen.0.clock
     3  real  OUT     0.8515425  siggen.0.cosine
     3  real  IN              1  siggen.0.frequency
     3  real  IN              0  siggen.0.offset
     3  real  OUT      2.772382  siggen.0.sawtooth
     3  real  OUT     -4.926954  siggen.0.sine
     3  real  OUT             5  siggen.0.square
     3  real  OUT      0.544764  siggen.0.triangle
     3  real  OUT     1.754e-06  siggen.0.update.time
     3  real  IO     1.6997e-05  siggen.0.update.tmax
     3  bool  OUT         FALSE  siggen.0.update.tmax-increased
```

Beachten Sie, dass sich der Wert des Parameters `siggen.0.amplitude` auf 5 geändert hat und dass die Stifte nun größere Werte haben.

Speichern der HAL-Konfiguration

Das meiste, was wir bisher mit `halcmd` gemacht haben, war einfach das Anzeigen von Dingen mit dem *show*-Befehl. Zwei der Befehle haben jedoch tatsächlich Dinge verändert. Wenn wir komplexere Systeme mit HAL entwerfen, werden wir viele Befehle verwenden, um die Dinge genau so zu konfigurieren, wie wir sie haben wollen. HAL hat ein Gedächtnis wie ein Elefant und behält diese Konfiguration bei, bis wir es abschalten. Aber was ist beim nächsten Mal? Wir wollen nicht jedes Mal, wenn wir das System benutzen wollen, eine Reihe von Befehlen manuell eingeben.

Speichern der Konfiguration des gesamten HAL mit einem einzigen Befehl.

```
halcmd: save

# components
loadrt threads name1=test-thread period1=1000000
loadrt siggen
# pin aliases
# signals
# nets
# parameter values
# realtime thread/function links
addf siggen.0.update test-thread
```

Die Ausgabe des Befehls `save` ist eine Folge von HAL-Befehlen. Wenn Sie mit einer *leeren* HAL beginnen und alle diese Befehle ausführen, erhalten Sie die Konfiguration, die zum Zeitpunkt der Ausgabe des Befehls *save* bestand. Um diese Befehle zur späteren Verwendung zu speichern, leiten wir die Ausgabe einfach in eine Datei um:

Speichern der Konfiguration in einer Datei mit `halcmd`

```
halcmd: save all saved.hal
```

Halrun beenden

Wenn Sie mit Ihrer HAL-Sitzung fertig sind, geben Sie an der Eingabeaufforderung `halcmd:`das Kommando `exit` ein. Damit kehren Sie zur System-Eingabeaufforderung zurück und beenden die HAL-Sitzung. Schließen Sie nicht einfach das Terminalfenster, ohne die HAL-Sitzung zu beenden.

HAL beenden

```
halcmd: exit
```

Wiederherstellung der HAL-Konfiguration

Um die in der Datei "saved.hal" gespeicherte HAL-Konfiguration wiederherzustellen, müssen wir alle diese HAL-Befehle ausführen. Dazu verwenden wir "`-f <Dateiname>'`" das die Befehle aus einer Datei liest, und "`-I`" (Großbuchstabe i), das die halcmd-Eingabeaufforderung nach Ausführung der Befehle anzeigt:

Ausführen einer gespeicherten Datei

```
halrun -I -f saved.hal
```

Beachten Sie, dass das Kommando "start" in saved.hal nicht vorhanden ist. Sie müssen dies erneut erteilen (oder die Datei saved.hal bearbeiten, um dies dort hinzuzufügen).

HAL aus dem Speicher entfernen

Wenn eine HAL-Sitzung unerwartet beendet wird, müssen Sie möglicherweise HAL entladen, bevor eine

neue Sitzung beginnen kann. Geben Sie dazu den folgenden Befehl in ein Terminalfenster ein.

Removing HAL

```
halrun -U
```

5.5.4. Stepgen Beispiel

Bis jetzt haben wir nur eine HAL-Komponente geladen. Die Idee hinter HAL ist jedoch, dass Sie eine Reihe von einfachen Komponenten laden und verbinden können, um ein komplexes System zu bilden. Das nächste Beispiel wird zwei Komponenten verwenden.

Bevor wir mit der Erstellung dieses neuen Beispiels beginnen können, wollen wir einen Neuanfang machen. Wenn Sie gerade eines der vorherigen Beispiele beendet haben, müssen wir alle Komponenten entfernen und die RTAPI- und HAL-Bibliotheken neu laden.

```
halcmd: exit
```

Installieren der Komponenten

Jetzt laden wir die Komponente Schrittimpulsgenerator. Eine detaillierte Beschreibung dieser Komponente finden Sie im Abschnitt stepgen des Integrator-Handbuchs. In diesem Beispiel verwenden wir den Steuertyp *velocity* von StepGen. Für den Moment können wir die Details überspringen und einfach die folgenden Befehle ausführen.

In diesem Beispiel wird der Kontrolltyp *velocity* aus der Komponente `stepgen` verwendet.

```
halrun
halcmd: loadrt stepgen step_type=0,0 ctrl_type=v,v
halcmd: loadrt siggen
halcmd: loadrt threads name1=fast period1=50000 name2=slow period2=1000000
```

The first command loads two step generators, both configured to generate stepping type 0. The second command loads our old friend siggen, and the third one creates two threads, a fast one with a period of 50 microseconds (µs) and a slow one with a period of 1 millisecond (ms).

NOTE The `fp1=` parameter is deprecated and ignored. All threads now unconditionally support floating point.

Wie zuvor können wir `halcmd show` verwenden, um einen Blick auf die HAL zu werfen. Diesmal haben wir viel mehr Pins und Parameter als zuvor:

```
halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     4  real  IN              1  siggen.0.amplitude
     4  bool  OUT         FALSE  siggen.0.clock
     4  real  OUT             0  siggen.0.cosine
```

```
     4  real  IN              1  siggen.0.frequency
     4  real  IN              0  siggen.0.offset
     4  real  OUT             0  siggen.0.sawtooth
     4  real  OUT             0  siggen.0.sine
     4  real  OUT             0  siggen.0.square
     4  real  OUT             0  siggen.0.triangle
     4  real  OUT             0  siggen.0.update.time
     4  real  IO              0  siggen.0.update.tmax
     4  bool  OUT         FALSE  siggen.0.update.tmax-increased
     3  sint  OUT             0  stepgen.0.counts
     3  bool  OUT         FALSE  stepgen.0.dir
     3  bool  IN          FALSE  stepgen.0.enable
     3  real  OUT             0  stepgen.0.position-fb
     3  bool  OUT         FALSE  stepgen.0.step
     3  real  IN              0  stepgen.0.velocity-cmd
     3  sint  OUT             0  stepgen.1.counts
     3  bool  OUT         FALSE  stepgen.1.dir
     3  bool  IN          FALSE  stepgen.1.enable
     3  real  OUT             0  stepgen.1.position-fb
     3  bool  OUT         FALSE  stepgen.1.step
     3  real  IN              0  stepgen.1.velocity-cmd
     3  real  OUT             0  stepgen.capture-position.time
     3  real  IO              0  stepgen.capture-position.tmax
     3  bool  OUT         FALSE  stepgen.capture-position.tmax-increased
     3  real  OUT             0  stepgen.make-pulses.time
     3  real  IO              0  stepgen.make-pulses.tmax
     3  bool  OUT         FALSE  stepgen.make-pulses.tmax-increased
     3  real  OUT             0  stepgen.update-freq.time
     3  real  IO              0  stepgen.update-freq.tmax
     3  bool  OUT         FALSE  stepgen.update-freq.tmax-increased

halcmd: show param

Parameters:
Owner   Type  Dir         Value  Name
     3  uint  RW     0x00000001  stepgen.0.dirhold
     3  uint  RW     0x00000001  stepgen.0.dirsetup
     3  real  RO              0  stepgen.0.frequency
     3  real  RW              0  stepgen.0.maxaccel
     3  real  RW              0  stepgen.0.maxvel
     3  real  RW              1  stepgen.0.position-scale
     3  sint  RO              0  stepgen.0.rawcounts
     3  uint  RW     0x00000001  stepgen.0.steplen
     3  uint  RW     0x00000001  stepgen.0.stepspace
     3  uint  RW     0x00000001  stepgen.1.dirhold
     3  uint  RW     0x00000001  stepgen.1.dirsetup
     3  real  RO              0  stepgen.1.frequency
     3  real  RW              0  stepgen.1.maxaccel
     3  real  RW              0  stepgen.1.maxvel
     3  real  RW              1  stepgen.1.position-scale
     3  sint  RO              0  stepgen.1.rawcounts
     3  uint  RW     0x00000001  stepgen.1.steplen
     3  uint  RW     0x00000001  stepgen.1.stepspace
```

Verbinden von Pins mit Signalen

Wir haben also zwei Schrittimpulsgeneratoren und einen Signalgenerator. Nun ist es an der Zeit, einige HAL-Signale zu erzeugen, um die beiden Komponenten zu verbinden. Wir tun so, als ob die beiden Schrittimpulsgeneratoren die X- und Y-Achse einer Maschine antreiben würden. Wir wollen den Tisch im Kreis bewegen. Dazu senden wir ein Kosinussignal an die X-Achse und ein Sinussignal an die Y-Achse. Das siggen-Modul erzeugt den Sinus und den Cosinus, aber wir brauchen "Drähte", um die Module miteinander zu verbinden. Im HAL werden diese "Drähte" Signale genannt. Wir müssen zwei davon erstellen. Wir können sie nennen, wie wir wollen, in diesem Beispiel werden sie *X-vel* und *Y-vel* heißen. Das Signal "X-vel" soll vom Cosinus-Ausgang des Signalgenerators zum Geschwindigkeitseingang des ersten Schrittimpulsgenerators führen. Der erste Schritt besteht darin, das Signal mit dem Ausgang des Signalgenerators zu verbinden. Um ein Signal mit einem Pin zu verbinden, verwenden wir den Netzbefehl.

net-Befehl

```
halcmd: net X-vel <= siggen.0.cosine
```

Um die Wirkung des Befehls `net` zu sehen, zeigen wir die Signale erneut.

```
halcmd: show sig

Signals:
Type          Value  Name     (linked to)
real              0  X-vel <== siggen.0.cosine
```

Wenn ein Signal mit einem oder mehreren Pins verbunden ist, listet der Befehl show die Pins unmittelbar nach dem Signalnamen auf. Der "Pfeil" zeigt die Richtung des Datenflusses an - in diesem Fall fließen die Daten vom Pin `siggen.0.cosine` zum Signal `X-vel`. Schließen wir nun das Signal `X-vel` an den Geschwindigkeitseingang eines Schrittimpulsgenerators an.

```
halcmd: net X-vel => stepgen.0.velocity-cmd
```

Wir können auch das Signal der Y-Achse `Y-vel` anschließen. Es soll vom Sinusausgang des Signalgenerators zum Eingang des zweiten Schrittimpulsgenerators laufen. Der folgende Befehl erreicht in einer Zeile, was zwei `net`-Befehle für `X-vel` erreicht haben.

```
halcmd: net Y-vel siggen.0.sine => stepgen.1.velocity-cmd
```

Werfen wir nun einen letzten Blick auf die Signale und die mit ihnen verbundenen Pins.

```
halcmd: show sig

Signals:
Type          Value  Name     (linked to)
real              0  X-vel <== siggen.0.cosine
                            ==> stepgen.0.velocity-cmd
real              0  Y-vel <== siggen.0.sine
                            ==> stepgen.1.velocity-cmd
```

Der Befehl *show sig* macht deutlich, wie genau die Daten durch den HAL fließen. Zum Beispiel kommt das *X-vel*-Signal von Pin `siggen.0.cosine` und geht zu Pin `stepgen.0.velocity-cmd`.

Einrichten der Echtzeitausführung - Threads und Funktionen

Wenn man sich vorstellt, dass Daten durch "Drähte" fließen, sind Pins und Signale recht einfach zu verstehen. Threads und Funktionen sind da schon etwas schwieriger. Funktionen enthalten die Computeranweisungen, welche die eigentliche Arbeit erledigen. Threads sind die Methode, mit der diese Anweisungen ausgeführt werden, wenn sie benötigt werden. Schauen wir uns zunächst die Funktionen an, die uns zur Verfügung stehen.

```
halcmd: show funct

Exported Functions:
Owner   CodeAddr  Arg       FP   Users  Name
 00004  f9992000  fc731278  YES      0   siggen.0.update
 00003  f998b20f  fc7310b8  YES      0   stepgen.capture-position
 00003  f998b000  fc7310b8  NO       0   stepgen.make-pulses
 00003  f998b307  fc7310b8  YES      0   stepgen.update-freq
```

Im Allgemeinen müssen Sie in der Dokumentation der einzelnen Komponenten nachschlagen, um zu erfahren, was ihre Funktionen bewirken. In diesem Fall wird die Funktion `siggen.0.update` verwendet, um die Ausgänge des Signalgenerators zu aktualisieren. Jedes Mal, wenn sie ausgeführt wird, berechnet sie die Werte der Sinus-, Kosinus-, Dreieck- und Quadratausgänge. Um glatte Signale zu erzeugen, muss sie in bestimmten Intervallen ausgeführt werden.

Die anderen drei Funktionen beziehen sich auf die Schrittimpulsgeneratoren.

Die erste, `stepgen.capture_position`, wird für die Positionsrückmeldung verwendet. Sie erfasst den Wert eines internen Zählers, der die Schrittimpulse zählt, während sie erzeugt werden. Unter der Annahme, dass keine Schritte ausgelassen werden, zeigt dieser Zähler die Position des Motors an.

The main function for the step pulse generator is `stepgen.make_pulses`. Every time *make_pulses* runs it decides if it is time to take a step, and if so sets the outputs accordingly. For smooth step pulses, it should run as frequently as possible. Because it needs to run so fast, *make_pulses* is highly optimized and performs only a few calculations.

Die letzte Funktion, `stepgen.update-freq`, ist für die Skalierung und einige andere Berechnungen zuständig, die nur durchgeführt werden müssen, wenn sich der Frequenzbefehl ändert.

Für unser Beispiel bedeutet dies, dass wir `siggen.0.update` mit einer moderaten Rate ausführen wollen, um die Sinus- und Kosinuswerte zu berechnen. Unmittelbar nachdem wir `siggen.0.update` ausgeführt haben, wollen wir `stepgen.update_freq` ausführen, um die neuen Werte in den Schrittimpulsgenerator zu laden. Schließlich müssen wir `stepgen.make_pulses` so schnell wie möglich ausführen, um gleichmäßige Impulse zu erhalten. Da wir keine Positionsrückmeldung verwenden, brauchen wir `stepgen.capture_position` überhaupt nicht auszuführen.

Wir führen Funktionen aus, indem wir sie zu Threads hinzufügen. Jeder Thread läuft mit einer bestimmten Geschwindigkeit. Schauen wir uns an, welche Threads wir zur Verfügung haben.

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     996980  YES                  slow (        0,        0 )
      49849  YES                  fast (        0,        0 )
```

The two threads were created when we loaded `threads`. The first one, *slow*, runs every millisecond. We will use it for `siggen.0.update` and `stepgen.update_freq`. The second thread is *fast*, which runs every 50 microseconds (µs). We will use it for `stepgen.make_pulses`. To connect the functions to the proper thread, we use the `addf` command. We specify the function first, followed by the thread.

```
halcmd: addf siggen.0.update slow
halcmd: addf stepgen.update-freq slow
halcmd: addf stepgen.make-pulses fast
```

Nachdem wir diese Befehle gegeben haben, können wir den Befehl `show thread` erneut ausführen, um zu sehen, was passiert ist.

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     996980  YES                  slow (        0,        0 )
                  1 siggen.0.update
                  2 stepgen.update-freq
      49849  NO                   fast (        0,        0 )
                  1 stepgen.make-pulses
```

Nun folgen auf jeden Thread die Namen der Funktionen in der Reihenfolge, in der sie ausgeführt werden sollen.

Parameter einstellen

Wir sind fast bereit, unser HAL-System zu starten. Allerdings müssen wir noch ein paar Parameter anpassen. Standardmäßig erzeugt die siggen-Komponente Signale, die von +1 bis -1 schwanken. Für unser Beispiel ist das in Ordnung, denn wir wollen, dass die Tischgeschwindigkeit zwischen +1 und -1 Zoll pro Sekunde schwankt. Die Skalierung des Schrittimpulsgenerators ist jedoch nicht ganz richtig. Standardmäßig erzeugt er eine Ausgangsfrequenz von 1 Schritt pro Sekunde bei einem Eingang von 1,0. Es ist unwahrscheinlich, dass ein Schritt pro Sekunde eine Tischbewegung von einem Zoll pro Sekunde ergibt. Nehmen wir stattdessen an, dass wir eine Leitspindel mit 5 Umdrehungen pro Zoll haben, die an einen Schrittmotor mit 200 Schritten pro Umdrehung und 10-fachem Mikroschritt angeschlossen ist. Eine Umdrehung der Spindel erfordert also 2000 Schritte und 5 Umdrehungen, um einen Zoll zu bewegen. Das bedeutet, dass die Gesamtskalierung 10000 Schritte pro Zoll beträgt. Wir müssen die Geschwindigkeitseingabe für den Schrittimpulsgenerator mit 10000 multiplizieren, um die richtige Ausgabe zu erhalten. Genau dafür ist der Parameter `stepgen.n.velocity-scale` gedacht. In diesem Fall haben sowohl die X- als auch die Y-Achse die gleiche Skalierung, also setzen wir die Skalierungsparameter für beide auf 10000.

```
halcmd: setp stepgen.0.position-scale 10000
halcmd: setp stepgen.1.position-scale 10000
halcmd: setp stepgen.0.enable 1
halcmd: setp stepgen.1.enable 1
```

Diese Geschwindigkeitssskalierung bedeutet, dass der Schrittgenerator 10000 Impulse pro Sekunde (10 kHz) erzeugt, wenn der Stift `stepgen.0.velocity-cmd` 1.0 beträgt. Bei der oben beschriebenen Motor- und Vortriebsschraube führt dies dazu, dass sich die Achse mit genau 1,0 Zoll pro Sekunde bewegt. Dies zeigt ein wichtiges, grundlegendes HAL-Konzept - Dinge wie Skalierung werden auf möglichst niedrigem Niveau durchgeführt, in diesem Fall im Schrittimpulsgenerator. Das interne Signal `X-vel` ist die Geschwindigkeit der Tabelle in Zoll pro Sekunde, und andere Komponenten wie `siggen` wissen überhaupt nicht (oder kümmern) über die Skalierung. Wenn wir die Leadcrew oder den Motor ändern, würden wir nur den Skalierungsparameter des Schrittimpulsgenerators ändern.

Ausführen!

Wir haben nun alles konfiguriert und können es starten. Genau wie im ersten Beispiel verwenden wir den Befehl `start`.

```
halcmd: start
```

Obwohl scheinbar nichts passiert, gibt der Schrittimpulsgenerator im Computer jede Sekunde Schrittimpulse aus, die von 10 kHz vorwärts bis 10 kHz rückwärts und wieder zurück reichen. Später in diesem Lernprogramm werden wir sehen, wie man diese internen Signale ausgibt, um Motoren in der realen Welt zu betreiben, aber zuerst wollen wir uns die Signale ansehen und sehen, was passiert.

5.5.5. Halmeter

Sie können sehr komplexe HAL-Systeme erstellen, ohne jemals eine grafische Oberfläche zu verwenden. Es hat jedoch etwas Befriedigendes, das Ergebnis seiner Arbeit zu sehen. Das erste und einfachste GUI-Werkzeug für HAL ist Halmeter. Es ist ein sehr einfaches Programm, das ein HAL-Äquivalent eines handlichen Multimeters darstellt.

Es ermöglicht die Beobachtung von Pins, Signalen oder Parametern, indem es den aktuellen Wert dieser Einheiten anzeigt. Es ist eine sehr einfach zu bedienende Anwendung für grafische Umgebungen. In einer Konsole geben Sie ein:

```
halmeter
```

Es erscheinen zwei Fenster. Das Auswahlfenster ist das größte und enthält drei Registerkarten:

- In der einen werden alle derzeit in HAL definierten Pins aufgelistet,
- eine Liste aller Signale,
- eine listet alle Parameter auf,

Klicken Sie auf eine Registerkarte und dann auf eines der Elemente, um es auszuwählen. In dem kleinen Fenster werden der Name und der Wert des ausgewählten Elements angezeigt. Die Anzeige wird etwa 10

Mal pro Sekunde aktualisiert. Um Platz auf dem Bildschirm zu schaffen, kann das Auswahlfenster mit der Schaltfläche *Close* geschlossen werden. Im kleinen Fenster, das beim Programmstart unter dem Auswahlfenster verborgen ist, öffnet die Schaltfläche *Auswählen* das Auswahlfenster erneut, und die Schaltfläche *Beenden* beendet das Programm und schließt beide Fenster.

Es ist möglich, mehrere Halmeter gleichzeitig laufen zu lassen, was die gleichzeitige Visualisierung mehrerer Elemente ermöglicht. Um ein Halmeter zu öffnen und die Konsole freizugeben, indem es im Hintergrund ausgeführt wird, führen Sie den folgenden Befehl aus:

```
halmeter &
```

Es ist möglich, halmeter zu starten und sofort ein Element anzeigen zu lassen. Fügen Sie dazu *pin|sig|par[am] name* Argumente in die Befehlszeile ein. Es wird das Signal, den Pin oder den Parameter *name* anzeigen, sobald es startet. Wenn das angegebene Element nicht vorhanden ist, wird es normal gestartet.

Wenn ein Element für die Anzeige angegeben wird, kann man *-s* vor pin|sig|param hinzufügen, um Halmeter anzuweisen, ein noch kleineres Fenster zu verwenden. Der Name des Elements wird dann in der Titelleiste statt unter dem Wert angezeigt, und es gibt keine Schaltfläche. Dies ist nützlich, wenn viele Halmeter auf kleinem Raum angezeigt werden sollen.

Wir werden erneut die Komponente siggen verwenden, um halmeter zu überprüfen. Wenn Sie das vorherige Beispiel gerade beendet haben, können Sie siggen mit der gespeicherten Datei laden. Wenn nicht, können wir es genauso laden wie zuvor:

```
halrun
halcmd: loadrt siggen
halcmd: loadrt threads name1=test-thread period1=1000000
halcmd: addf siggen.0.update test-thread
halcmd: start
halcmd: setp siggen.0.amplitude 5
```

Zu diesem Zeitpunkt ist die Komponente siggen geladen und läuft. Es ist an der Zeit, halmeter zu starten.

Halmeter starten

```
halcmd: loadusr halmeter
```

Das erste Fenster, das Sie sehen, ist das Fenster "Zu untersuchendes Element auswählen" (engl. select Item to probe).

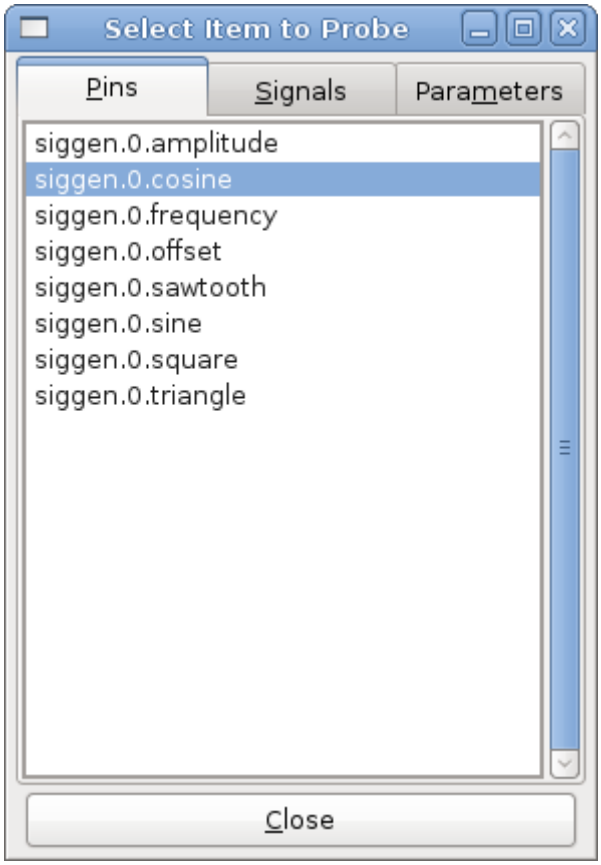


Figure 73. Halmeter Auswahlfenster

Dieser Dialog hat drei Registerkarten. Auf der ersten Registerkarte werden alle HAL-Pins des Systems angezeigt. Auf der zweiten Registerkarte werden alle Signale angezeigt, und auf der dritten alle Parameter. Wir möchten uns zuerst den Pin `siggen.0.cosine` ansehen, also klicken Sie darauf und dann auf die Schaltfläche "Schließen" (engl. close). Das Dialogfeld für die Sondenauswahl wird geschlossen, und das Messgerät sieht ungefähr so aus wie in der folgenden Abbildung.



Figure 74. Halmeter-Fenster

Um zu ändern, was das Messgerät anzeigt, drücken Sie auf die Schaltfläche "Auswählen", wodurch das Fenster "Zu messendes Element auswählen" wieder angezeigt wird.

Sie sollten sehen, wie sich der Wert ändert, wenn siggen seine Kosinuswelle erzeugt. Das Halmeter aktualisiert seine Anzeige etwa 5 Mal pro Sekunde.

Zum Beenden von Halmeter klicken Sie einfach auf die Schaltfläche Beenden.

Wenn Sie mehr als einen Pin, ein Signal oder einen Parameter auf einmal betrachten wollen, können Sie einfach mehrere Halmeter starten. Das Halmeter-Fenster wurde absichtlich sehr klein gehalten, damit Sie viele davon gleichzeitig auf dem Bildschirm haben können.

5.5.6. Halshow

Das Skript `halshow` kann Ihnen helfen, sich in einer laufenden HAL zurechtzufinden. Es zeigt ausgewählte HAL-Werte an und aktualisiert sie laufend.

Es handelt sich um ein sehr spezialisiertes System, das mit einem funktionierenden HAL verbunden sein muss. Es kann nicht eigenständig laufen, weil es sich auf die Fähigkeit von HAL verlässt, in sich hineinzuhören (engl. introspect) und berichten, was es von sich selbst durch die halcmd Interface-Bibliothek weiß. Wenn sich die Konfiguration von LinuxCNC ändert, wird auch die Ausgabe von halshow anders sein.

Wie wir gleich sehen werden, ist diese Fähigkeit von HAL, sich selbst zu dokumentieren, ein Schlüssel für den Aufbau eines effektiven CNC-Systems.

Halshow starten

Halshow Ist verfügbar

- im AXIS-Menü unter Maschine/Zeige HAL-Konfiguration,
- im TkLinuxCNC-Menü unter Scripts/HAL Show,
- in GMOCCAPY auf der Seite mit Einstellungen.

`halshow` can also be started from a terminal command line and specify formats for integer and real items (pins or signals) and identify a saved watchlist file to use:

```
$ halshow --help
Usage:
  halshow [Options] [watchfile]
  Options:
           --help    (this help)
           --fformat format_string_for_float
           --iformat format_string_for_int
           --noprefs don't use preference file to save settings

Hinweise:
       Erstellen Sie einen watchfile in halshow mit: 'File/Save Watch List'.
       LinuxCNC muss für die Standalone-Nutzung ausgeführt werden.
```

Beispiel für die Begrenzung der Anzahl der Dezimalpunkte für Fließkommazahlen und die Verwendung einer Datei namens my.halshow im aktuellen Verzeichnis:

```
$ halshow --fformat "%.5f" ./my.halshow
```

For more information regarding the format, please refer to the Tcl format man page.

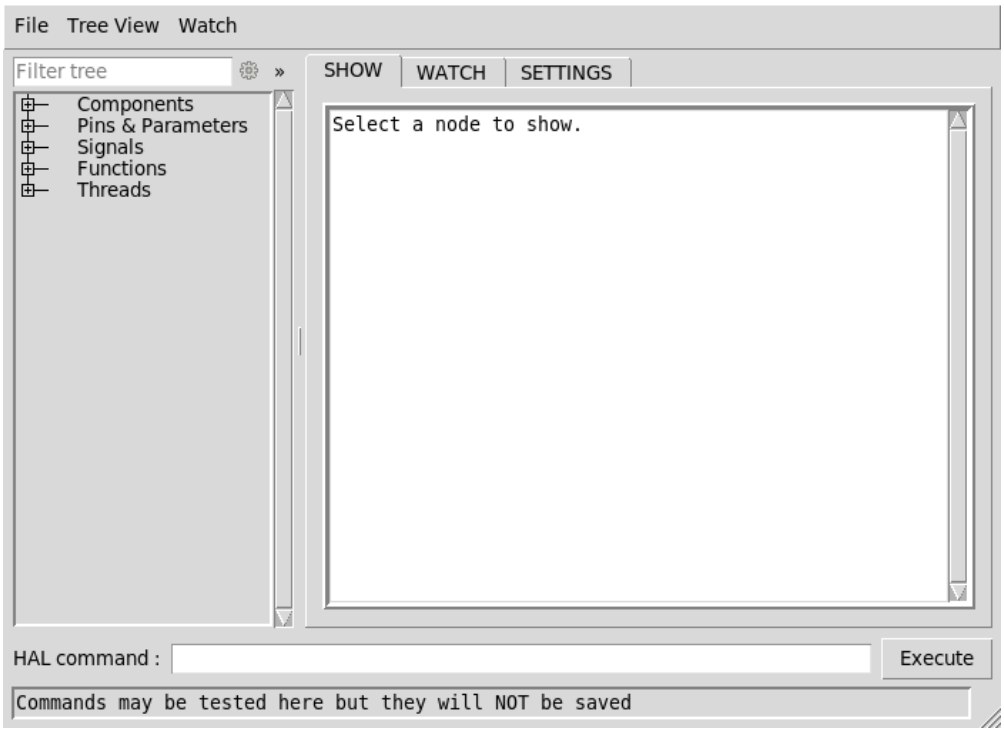


Figure 75. Halshow-Layout

Auf der linken Seite des Bildschirms, wie in der obigen Abbildung gezeigt, befindet sich eine Baumansicht, die dem Dateibrowser ähnelt, den Sie vielleicht kennen. Auf der rechten Seite befindet sich ein Notizbuch mit Registerkarten für Anzeigen, Beobachten und Einstellungen.

HAL Baum-Darstellung (engl. Tree Area)

Filter-Baum

Standardmäßig filtert dieser Eintrag den Baum nach Pin-Namen oder Baum-Knoten durch einen regulären Ausdruck. Zum Beispiel würde die Eingabe von "lim-sw" den Baum nach dem folgenden filtern:

Filterbaum für Pin-Namen mit der Teilzeichenkette "lim-sw".

```
joint.0.neg-lim-sw-in
joint.0.pos-lim-sw-in
joint.1.neg-lim-sw-in
joint.1.pos-lim-sw-in
joint.2.neg-lim-sw-in
joint.3.pos-lim-sw-in
```

Wenn Sie alle "joint.0"-Beziehungen anzeigen möchten, müssen Sie auf das Einstellungssymbol klicken und "Full path" (engl. für vollständiger Pfad) auswählen. Achten Sie darauf, die Sonderzeichen für reguläre Ausdrücke zu escapen, d.h. Sie müssen "joint\.1\." eingeben, um explizit den Punkt abzufragen und nicht auch joint 10 zu finden.

Baum

Der Baum zeigt alle wichtigen Teile eines HAL. Vor jedem Teil befindet sich ein kleines Plus- (+) oder Minuszeichen (-) in einem Kästchen, mit dem Sie den entsprechenden Teil des Baums auf- oder zuklappen können.

Sie können die Baumansicht auch über das Menü Baumansicht (engl. tree view) am oberen linken Rand der Anzeige erweitern oder reduzieren.

Unter der *Baumansicht* finden Sie: *Alle aufklappen* (engl. expand all), *Alle zuklappen* (engl. collapse all); *Pins aufklappen* (engl. expand pins), *Parameter aufklappen* (engl. expand parameters), *Signale aufklappen* (engl. expand signals); und *Baumansicht neu laden* (engl. reload tree view). *Baumansicht neu laden* ist nützlich, wenn zur Laufzeit neue Komponenten geladen werden und angezeigt werden sollen.

Because the pins and parameters of a component cannot share the same name, they are listed together in a single *Pins & Parameters* branch, with parameters shown in brown. To list pins and parameters in separate branches, enable *Separate parameters from pins in tree* in the Settings tab.

HAL Show Area

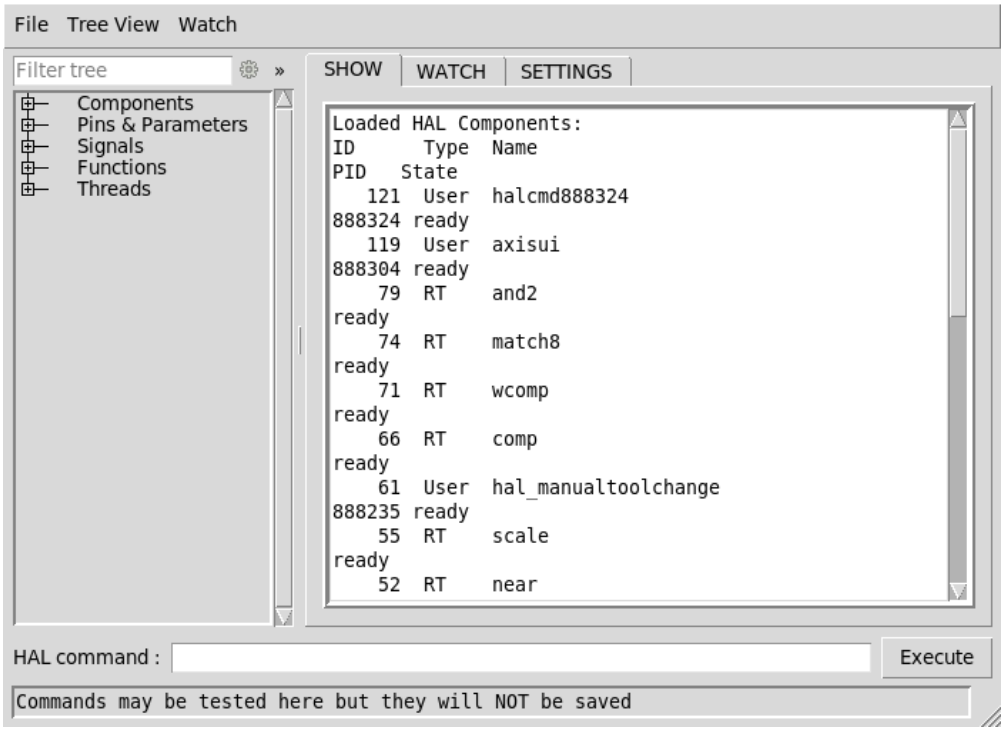


Figure 76. Halshow: Show Tab (engl. für Registerkarte anzeigen)

Wenn Sie auf den Namen des Knotens klicken, z.B. "Komponenten" (engl. components) im Baum, wird Ihnen (unter der Registerkarte "Show") alles angezeigt, was HAL über den Inhalt dieses Knotens weiß. Abbildung Halshow Show Tab zeigt eine Liste, die genau so aussieht, wie wenn Sie auf den Namen "Components" klicken. Die Informationsanzeige ist genau wie bei den traditionellen textbasierten HAL-Analysetools. Der Vorteil hier ist, dass wir per Mausklick Zugriff haben, und zwar so umfassend oder so gezielt, wie Sie es brauchen.

Wenn wir uns die Baumdarstellung genauer ansehen, sehen wir, dass die sechs Hauptteile eines HAL alle um mindestens eine Ebene erweitert werden können. Wenn diese Ebenen erweitert werden, können Sie die Antwort genauer betrachten, wenn Sie auf den ganz rechten Baumknoten klicken. Sie werden feststellen, dass es einige HAL-Pins und Parameter gibt, die mehr als eine Antwort anzeigen. Dies liegt an der Art der Suchroutinen in halcmd selbst. Wenn Sie einen Pin suchen, erhalten Sie möglicherweise zwei Antworten, wie hier:

```
Component Pins:
Owner  Type  Dir  Value  Name
06     bool   -W   TRUE  parport.0.pin-10-in
06     bool   -W  FALSE  parport.0.pin-10-in-not
```

Der Name des zweiten Pins enthält den vollständigen Namen des ersten.

Ausgewählter Text auf der Registerkarte Anzeigen kann mit der rechten Maustaste oder STRG-C kopiert werden.
Im Bereich Anzeigen können Sie Stecknadeln aus ausgewähltem Text hinzufügen, indem Sie das Kontextmenü verwenden. Alle gültigen Pins, die in der Auswahl enthalten sind, werden der Registerkarte "Beobachten" (engl. watch) hinzugefügt.

Watch Area

Wenn Sie auf die Registerkarte "Beobachten" (engl. watch) klicken, wird eine leere Leinwand (engl. canvas) angezeigt. Sie können Signale und Pins zu diesem Canvas hinzufügen und ihre Werte beobachten. Sie können Signale oder Pins hinzufügen, wenn die Registerkarte "Überwachen" angezeigt wird, indem Sie in der Strukturansicht auf den Namen des Signals klicken.

Sie können auch alle Unterpunkte dieses Knotens hinzufügen, indem Sie diese im Rechtsklickmenü auswählen (siehe Abbildung Halshow Watch Tab).

Die folgende Abbildung zeigt diese Leinwand (engl. canvas) mit mehreren Pins.

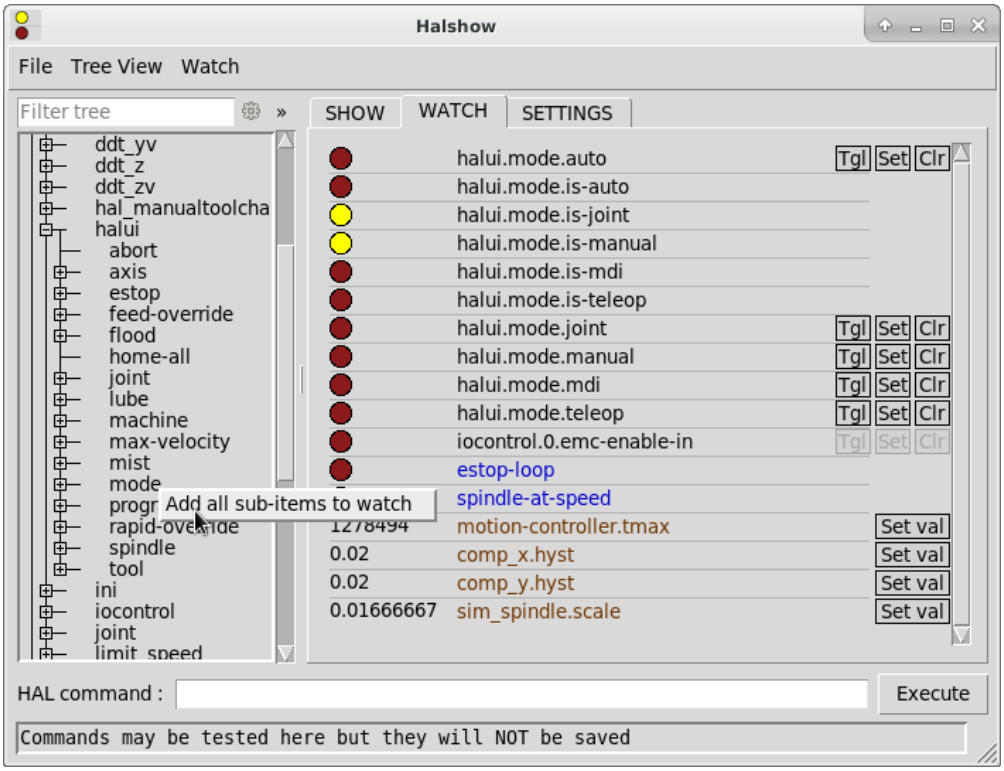


Figure 77. Halshow: Watch-Tab

Watch displays bool type (binary) values using colored circles representing LEDs. They show as dark red

when a bool signal or pin is false, and as light yellow whenever that signal is true. If you select a pin or signal that is not a bool type (binary) signal, *watch* will show it as a numerical value. Pins are displayed in black, signals in blue and parameters in brown.

Watch ermöglicht es Ihnen, schnell Schalter zu testen oder die Auswirkungen von Änderungen zu sehen, die Sie an LinuxCNC vornehmen, während Sie die grafische Benutzeroberfläche verwenden. Die Bildwiederholfrequenz von *watch* ist etwas zu langsam, um Stepperimpulse zu sehen, aber Sie können sie für diese verwenden, wenn Sie eine Achse sehr langsam oder in sehr kleinen Entfernungsschritten bewegen.

Die Pins und Signale, die beschreibbar sind, haben auf der rechten Seite Schaltflächen zur Manipulation. Pins, die mit einem Signal verknüpft sind, haben deaktivierte Schaltflächen. Um diese Werte zu setzen, muss der entsprechende Pin vom Signal entkoppelt werden. Dies kann durch Rechtsklick auf den Signalnamen und Auswahl von "Unlink pin" geschehen, siehe Watch Tach Context Menu.

Die Beobachtungsliste wird beim Beenden automatisch gespeichert. Wenn du nicht möchtest, dass Halshow deine Beobachtungsliste speichert, kannst du dies in den Einstellungen (engl. settings) deaktivieren.

Kontextmenü

Das Kontextmenü ermöglicht zudem:

- Kopieren des Pinnamens in die Zwischenablage
- Einen Wert einstellen
- Einen Pin trennen (sofern mit einem Signal verbunden)
- Apin in der Strukturansicht anzeigen (hebt den Pin hervor, scrollt nicht zur Position)
- Entfernen eines Pins aus der Liste

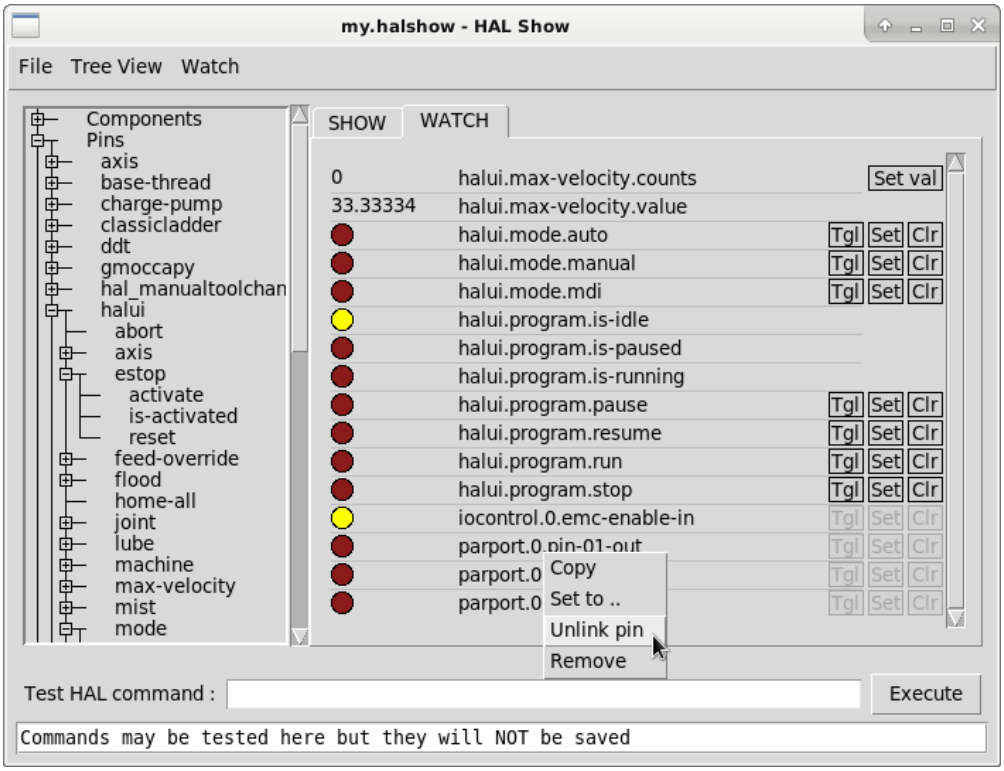


Figure 78. Halshow: Kontextmenü der Registerkarte Watch

Befehls-Eingabe

Im unteren Teil befindet sich ein Eingabefeld zum Testen von HAL-Befehlen. Die Befehle, die Sie hier eingeben und die Auswirkungen, die sie auf die laufende HAL haben, werden nicht gespeichert. Sie bleiben erhalten, solange LinuxCNC läuft, werden aber gelöscht, sobald LinuxCNC nicht mehr läuft.

Die Befehlseingabe hat einee BASH-ähnliche Historie (während der Sitzung), so dass Sie eingefügte Befehle mit der Pfeiltaste nach oben wiederherstellen können.

Das Eingabefeld "HAL Command:" akzeptiert jeden der für halcmd aufgeführten Befehle. Dazu gehören:

- `loadrt`, `unloadrt` (Laden/Entladen von Echtzeitmodulen)
- loadusr", "unloadusr" (Laden/Entladen von Nicht-Echtzeit-Komponenten)
- `addf`, `delf` (Hinzufügen/Löschen einer Funktion zu/aus einem Echtzeit-Thread)
- `net` (eine Verbindung zwischen zwei oder mehr Elementen herstellen)
- `setp` (Parameter (oder Pin) auf einen Wert setzen)

Dieser kleine Editor gibt jedes Mal einen Befehl ein, wenn Sie *Enter* drücken oder die Schaltfläche Ausführen betätigen. Eine Fehlermeldung von halcmd wird angezeigt, wenn diese Befehle nicht richtig gebildet sind. Wenn Sie nicht sicher sind, wie Sie einen korrekten Befehl eingeben, müssen Sie die Dokumentation zu halcmd und den spezifischen Modulen, mit denen Sie arbeiten, erneut lesen.

Einstellungen

The geometry of the window and the settings are saved in a file in the configuration directory on exit. If that path cannot be determined, they are stored in the home directory. The path will be displayed in the

settings page. You can omit using the preferences file by calling halshow with the command line argument `--no-prefs`.

Override format string

Sets the format of the displayed values, for more information refer to the Tcl format man page.

Always on top

Keeps the halshow window on top.

Remember watchlist

Keeps the saved items in the watchlist on exit.

Separate parameters from pins in tree

Lists pins and parameters in separate *Pins* and *Parameters* branches of the tree, instead of the combined *Pins & Parameters* branch. The setting takes effect when you press *Apply* and is remembered between sessions.

The further settings should be self-explaining.

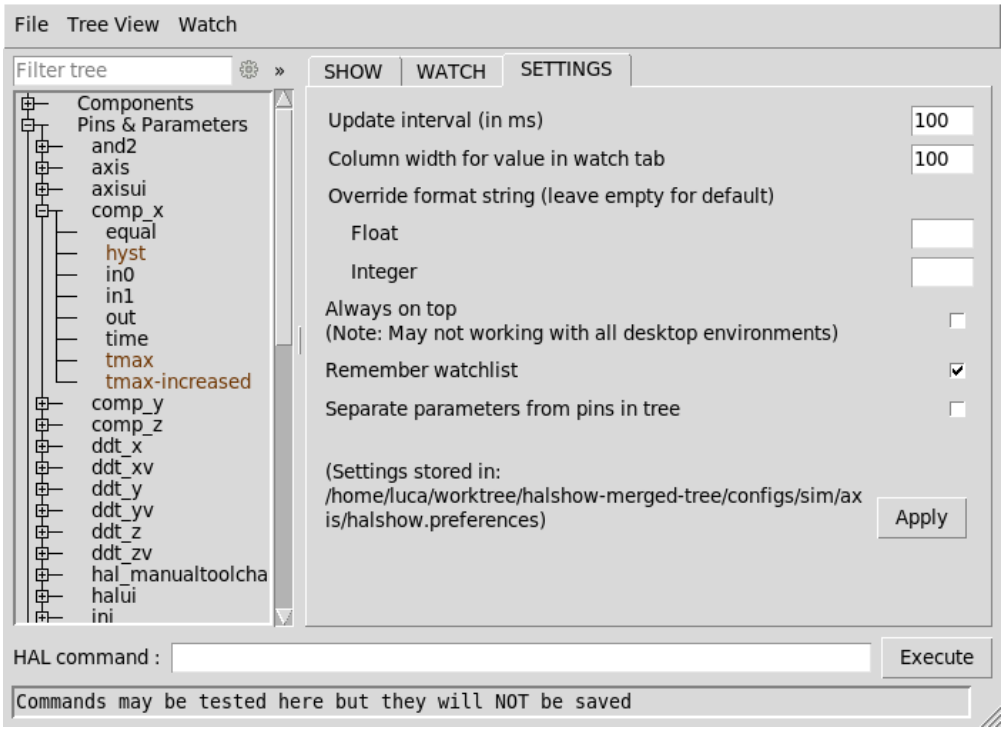


Figure 79. Halshow Einstellungen

Beispiel/Tutorial

Lassen Sie uns diesen Editor verwenden, um ein Differenzierungs-Modul (engl. differential module) zu einem HAL hinzuzufügen und es mit der Achsenposition zu verbinden, damit wir die Änderungsrate der Position, d.h. die Beschleunigung, sehen können. Zunächst müssen wir eine HAL-Komponente mit dem Namen ddt laden, sie zum Servo-Thread hinzufügen und sie dann mit dem Positionsstift eines Gelenks verbinden. Danach können wir den Ausgang des Differenzierers in Halscope finden. Also los geht's.

```
loadrt ddt
```

Schauen Sie sich den Komponenten-Knoten an, und Sie sollten dort irgendwo ddt finden.

```
Geladene HAL Komponenten:
ID Type        Name
10 User halcmd29800
09 User halcmd29374
08   RT         ddt
06   RT hal_parport
05   RT    scope_rt
04   RT     stepgen
03   RT      motmod
02 User   iocontrol
```

Und tatsächlich, da ist sie. Beachten Sie, dass seine ID 08 ist. Als Nächstes müssen wir herausfinden, welche Funktionen damit verfügbar sind, also schauen wir uns die Funktionen an:

```
Exportierte Funktionen:
Owner  CodeAddr      Arg  FP Users Name
  08   E0B97630 E0DC7674 YES     0 ddt.0
  03   E0DEF83C 00000000 YES     1 motion-command-handler
  03   E0DF0BF3 00000000 YES     1 motion-controller
  06   E0B541FE E0DC75B8  NO     1 parport.0.read
  06   E0B54270 E0DC75B8  NO     1 parport.0.write
  06   E0B54309 E0DC75B8  NO     0 parport.read-all
  06   E0B5433A E0DC75B8  NO     0 parport.write-all
  05   E0AD712D 00000000  NO     0 scope.sample
  04   E0B618C1 E0DC7448 YES     1 stepgen.capture-position
  04   E0B612F5 E0DC7448  NO     1 stepgen.make-pulses
  04   E0B614AD E0DC7448 YES     1 stepgen.update-freq
```

Hier suchen wir nach Eigentümer #08 und sehen eine Funktion namens `ddt.0`. Wir sollten in der Lage sein, `ddt.0` zum Servo-Thread hinzuzufügen, und es wird seine Berechnungen jedes Mal durchführen, wenn der Servo-Thread aktualisiert wird. Noch einmal sehen wir uns den addf-Befehl an und stellen fest, dass er drei Argumente wie folgt verwendet:

```
addf <Funktionsname> <Threadname> [<Position>]
```

Wir wissen bereits, dass Funktionsname=ddt.0 ist, also müssen wir den Thread-Namen richtig eingeben, indem wir den Thread-Knoten im Baum erweitern. Hier sehen wir zwei Threads, den Servo-Thread und den Basis-Thread. Die Position von `ddt.0` im Thread ist nicht entscheidend. Wir fügen also die Funktion `ddt.0` zum Servo-Thread hinzu:

```
addf ddt.0 servo-thread
```

Dies ist nur zur Ansicht, also lassen wir Position leer und erhalten die letzte Position im Thread. Die folgende Abbildung zeigt den Zustand von halshow, nachdem dieser Befehl erteilt wurde.

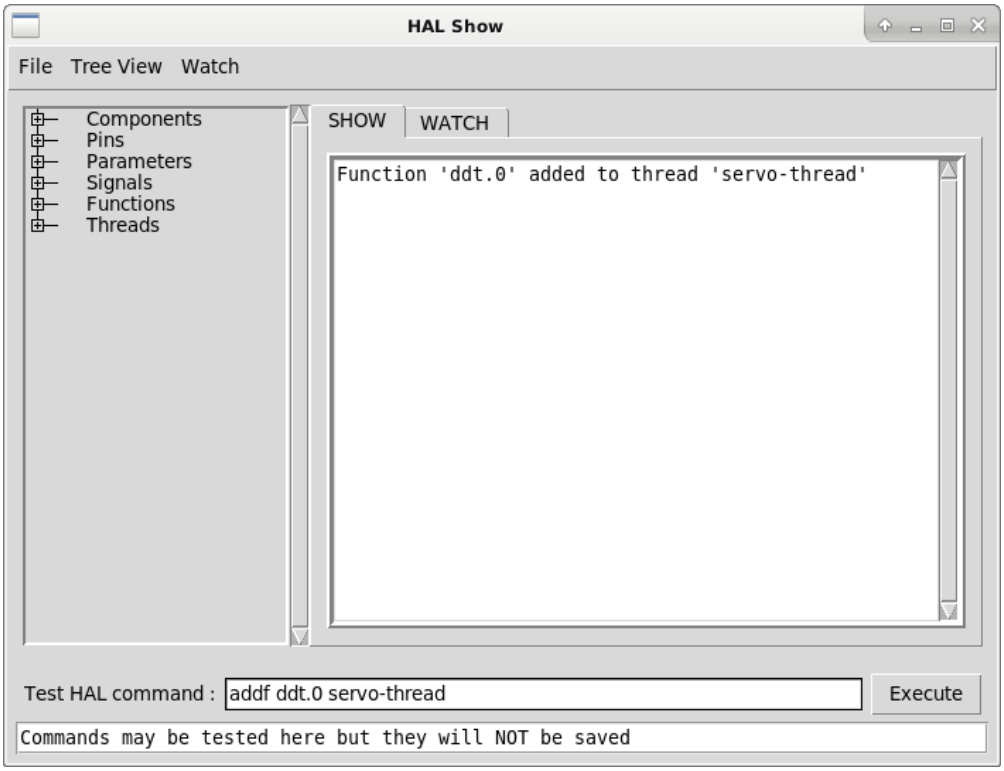


Figure 80. Addf-Befehl

Als nächstes müssen wir ddt mit etwas verbinden. Aber woher wissen wir, welche Pins verfügbar sind? Die Antwort ist, dass wir unter Pins nachsehen. Dort finden wir `ddt` und sehen dies:

```
Component Pins:
Owner Type  Dir Value       Name
08    real  R-  0.00000e+00 ddt.0.in
08    real  -W  0.00000e+00 ddt.0.out
```

Das sieht einfach genug aus, um es zu verstehen, aber welches Signal oder welchen Pin wollen wir damit verbinden? Es könnte ein Achsen-Pin, ein Stepgen-Pin oder ein Signal sein. Wir sehen dies, wenn wir uns `joint.0` ansehen:

```
Component Pins:
Owner Type  Dir Value       Name
03    real  -W  0.00000e+00 joint.0.motor-pos-cmd ==> Xpos-cmd
```

Es sieht also so aus, als ob Xpos-cmd ein gutes Signal sein sollte. Zurück zum Editor, wo wir den folgenden Befehl eingeben:

```
linksp Xpos-cmd ddt.0.in
```

Wenn wir uns nun das `Xpos-cmd`-Signal mithilfe des Baumknotens ansehen, sehen wir, was wir getan haben:

```
Signals:
Type Value Name
real  0.00000e+00 Xpos-cmd
<== joint.0.motor-pos-cmd
```

```
==> ddt.0.in
==> stepgen.0.position-cmd
```

Wir sehen, dass dieses Signal von `joint.0.motor-pos-cmd` kommt und sowohl an `ddt.0.in` als auch an `stepgen.0.position-cmd` geht. Durch die Verbindung unseres Blocks mit dem Signal haben wir jegliche Komplikationen mit dem normalen Ablauf dieses Bewegungsbefehls vermieden.

Der HAL-Anzeigebereich (engl. show area) verwendet `halcmd`, um herauszufinden, was in einermlaufenden HAL passiert. Er gibt Ihnen vollständige Informationen darüber, was er entdeckt hat. Er wird auch aktualisiert, wenn Sie über das kleine Editor-Panel Befehle zur Änderung des HAL eingeben. Es kann vorkommen, dass Sie andere Dinge angezeigt haben möchten, ohne dass alle Informationen in diesem Bereich verfügbar sind. In diesem Fall ist der HAL-Überwachungsbereich von Nutzen.

5.5.7. Halscope

Das vorherige Beispiel erzeugt einige sehr interessante Signale. Aber vieles von dem, was passiert, ist viel zu schnell, um es mit dem Halmeter zu sehen. Um einen genaueren Blick auf die Vorgänge im Inneren des HAL zu werfen, brauchen wir ein Oszilloskop. Glücklicherweise verfügt HAL über ein solches, genannt halscope.

Halscope besteht aus zwei Teilen - einem Echtzeit-Teil, der die HAL-Signale liest, und einem Nicht-Echtzeit-Teil, der die grafische Benutzeroberfläche und die Anzeige bereitstellt. Sie müssen sich jedoch nicht darum kümmern, da der Nicht-Echtzeit-Teil den Echtzeit-Teil bei Bedarf automatisch lädt.

Wenn LinuxCNC in einem Terminal läuft, können Sie halscope mit dem folgenden Befehl starten.

Halscope starten

```
halcmd loadusr halscope
```

Wenn LinuxCNC nicht läuft oder die Datei autosave.halscope nicht mit den Pins übereinstimmt, die im aktuell laufenden LinuxCNC verfügbar sind, öffnet sich das Scope-GUI-Fenster, unmittelbar gefolgt von einem Dialog *Realtime function not linked*, der wie die folgende Abbildung aussieht. Um die Abtastrate zu ändern, klicken Sie mit der linken Maustaste auf das Feld Samples.

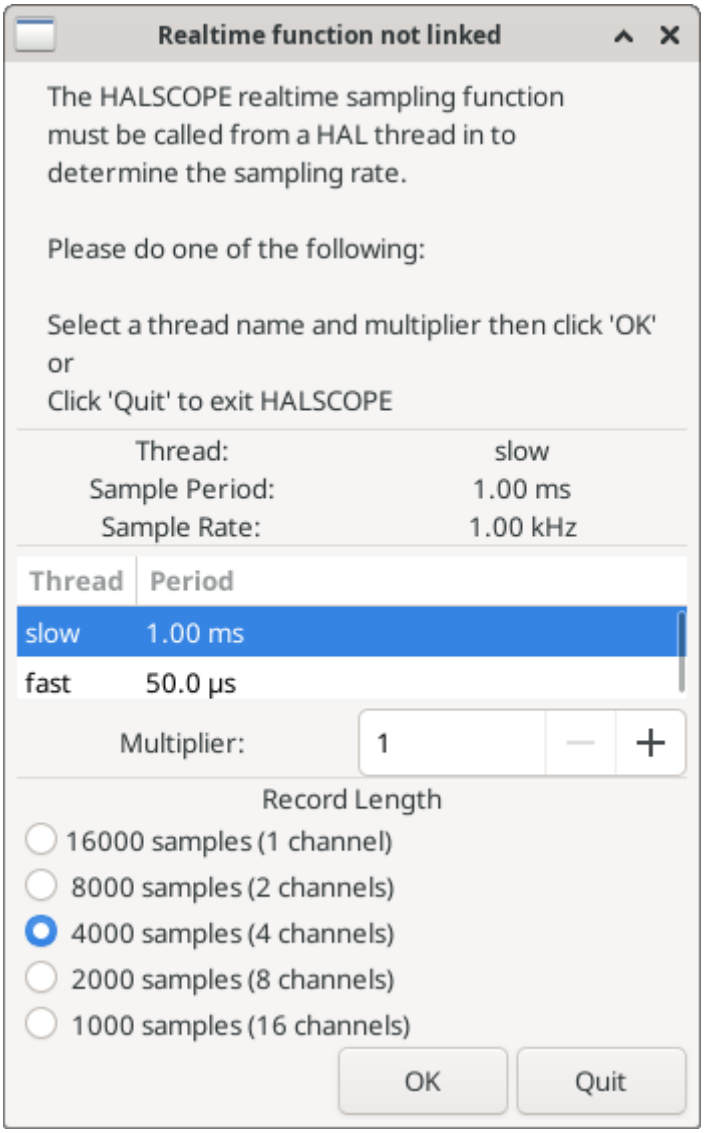


Figure 81. Dialog Echtzeitfunktion nicht verknüpft

In diesem Dialogfeld stellen Sie die Abtastrate für das Oszilloskop ein. Für den Moment wollen wir einmal pro Millisekunde abtasten, also klicken wir auf den 1 ms-Thread *slow* und lassen den Multiplikator bei 1. Wir werden auch die Aufzeichnungslänge bei 4000 Abtastungen belassen, so dass wir bis zu vier Kanäle auf einmal verwenden können. Wenn Sie einen Thread auswählen und dann auf "OK" klicken, verschwindet das Dialogfeld, und das Scope-Fenster sieht ungefähr so aus wie in der folgenden Abbildung.

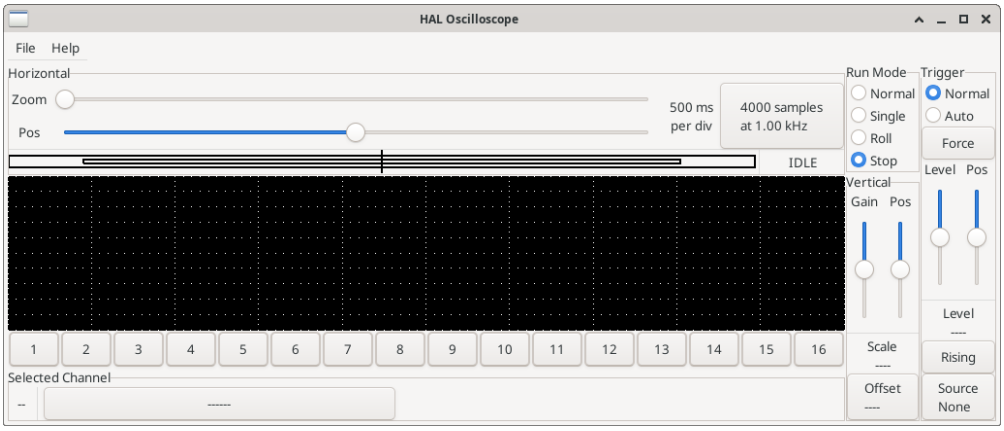


Figure 82. Fenster für den anfänglichen Geltungsbereich

Anschließen der Oszilloskop-Sonden

An diesem Punkt ist Halscope einsatzbereit. Wir haben bereits eine Abtastrate und eine Aufzeichnungslänge gewählt, so dass der nächste Schritt darin besteht, zu entscheiden, was wir uns ansehen wollen. Dies ist gleichbedeutend mit dem Anschließen von "virtuellen Oszilloskop-Sonden" an den HAL. Halscope verfügt über 16 Kanäle, aber die Anzahl, die Sie gleichzeitig verwenden können, hängt von der Aufzeichnungslänge ab - mehr Kanäle bedeuten kürzere Aufzeichnungen, da der für die Aufzeichnung verfügbare Speicher auf etwa 16.000 Samples festgelegt ist.

Die Kanalschaltflächen befinden sich am unteren Rand des Halskop-Bildschirms. Wenn Sie auf die Schaltfläche "1" klicken, wird das Dialogfeld "Select Channel Source" (Kanalquelle auswählen) angezeigt, wie in der folgenden Abbildung dargestellt. Dieser Dialog ist dem von Halmeter verwendeten Dialog sehr ähnlich. Wir möchten uns die Signale ansehen, die wir zuvor definiert haben, also klicken wir auf die Registerkarte "Signale", und der Dialog zeigt alle Signale im HAL an (in diesem Beispiel nur zwei).

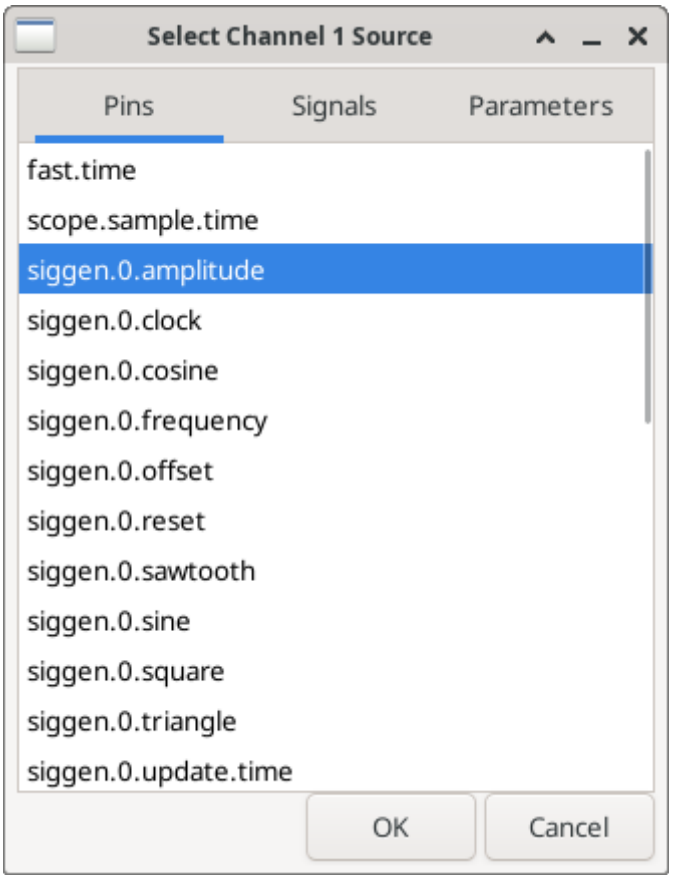


Figure 83. Kanalquelle auswählen

Um ein Signal auszuwählen, klicken Sie es einfach an. In diesem Fall möchten wir, dass auf Kanal 1 das Signal "X-vel" angezeigt wird. Klicken Sie auf die Registerkarte "Signale" und dann auf "X-vel". Das Dialogfeld schließt sich und der Kanal ist nun ausgewählt.

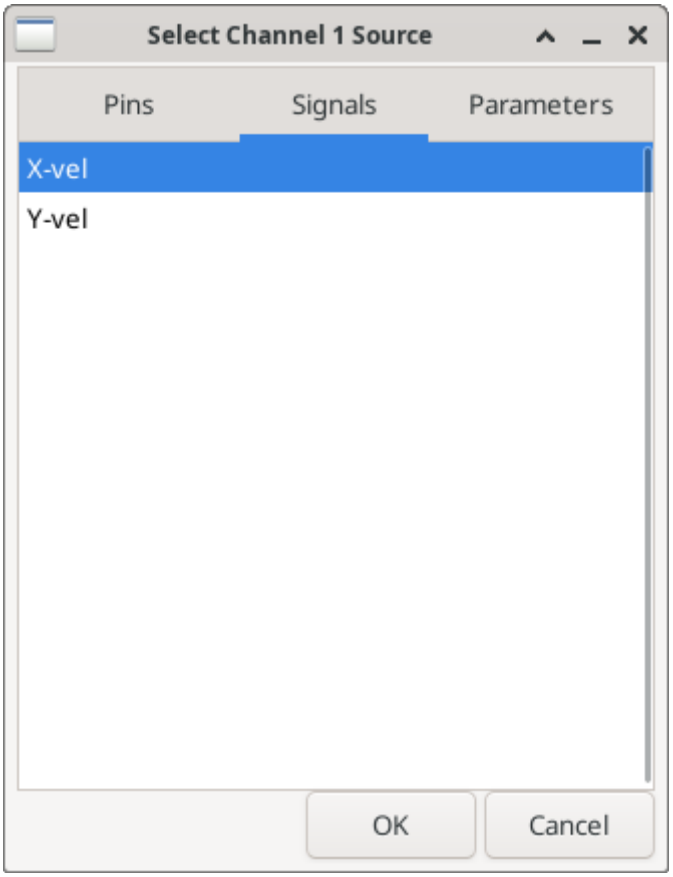


Figure 84. Signal auswählen

Die Taste für Kanal 1 wird gedrückt, und die Kanalnummer 1 und die Bezeichnung "X-vel" erscheinen unter der Tastenreihe. Diese Anzeige zeigt immer den ausgewählten Kanal an - Sie können mehrere Kanäle auf dem Bildschirm haben, aber der ausgewählte Kanal ist hervorgehoben, und die verschiedenen Steuerelemente wie vertikale Position und Skalierung funktionieren immer für den ausgewählten Kanal.

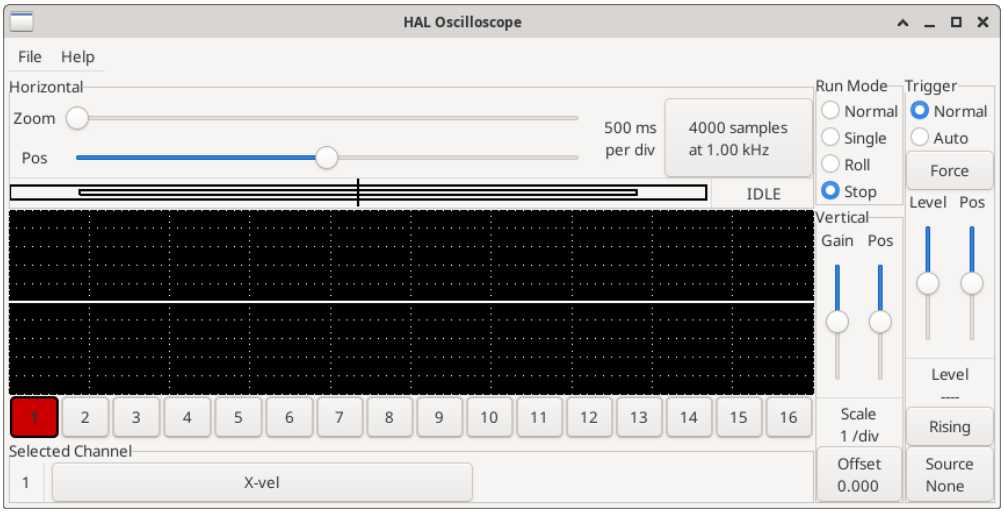


Figure 85. Halscope

Um ein Signal zu Kanal 2 hinzuzufügen, klicken Sie auf die Schaltfläche "2". Wenn der Dialog angezeigt wird, klicken Sie auf die Registerkarte "Signale" und dann auf "Y-vel". Wir wollen uns auch die Rechteck- und Dreieckswellenausgänge ansehen. Es sind keine Signale mit diesen Pins verbunden, daher verwenden wir stattdessen die Registerkarte "Pins". Für Kanal 3 wählen Sie `siggen.0.triangle` und

für Kanal 4 `siggen.0.square`.

Erfassen unserer ersten Wellenformen

Nachdem wir nun mehrere Sonden an den HAL angeschlossen haben, ist es an der Zeit, einige Wellenformen zu erfassen. Zum Starten des Oszilloskops klicken Sie auf die Schaltfläche "Normal" im Abschnitt "Run Mode" des Bildschirms (oben rechts). Da wir eine Aufzeichnungslänge von 4000 Samples haben und 1000 Samples pro Sekunde erfassen, wird halscope etwa 2 Sekunden brauchen, um die Hälfte seines Puffers zu füllen. Während dieser Zeit zeigt ein Fortschrittsbalken direkt über dem Hauptbildschirm an, dass der Puffer gefüllt ist. Sobald der Puffer halb voll ist, wartet das Scope auf einen Trigger. Da wir noch keinen konfiguriert haben, wird es ewig warten. Um es manuell auszulösen, klicken Sie auf die Schaltfläche "Erzwingen" im Abschnitt "Auslöser" oben rechts. Sie sollten sehen, wie sich der Rest des Puffers füllt, und dann werden die erfassten Wellenformen auf dem Bildschirm angezeigt. Das Ergebnis sieht ungefähr so aus wie in der folgenden Abbildung.

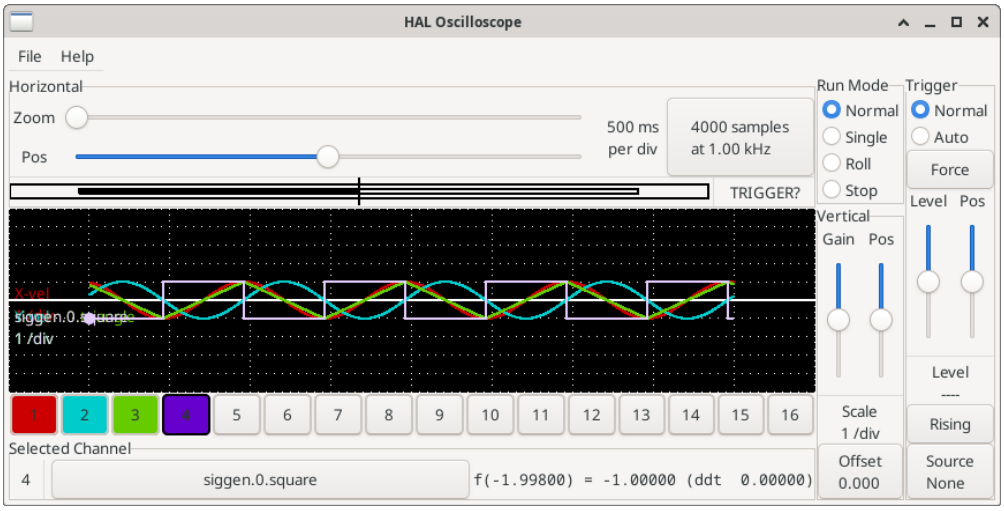


Figure 86. Erfasste Wellenformen

Das Feld *Ausgewählter Kanal* am unteren Rand zeigt an, dass die violette Kurve die aktuell ausgewählte Kurve ist, Kanal 4, der den Wert des Pins `siggen.0.square` anzeigt. Klicken Sie auf die Kanalschaltflächen 1 bis 3, um die anderen drei Spuren zu markieren.

Vertikale Anpassungen

Die Spuren sind nur schwer zu unterscheiden, da alle vier übereinander liegen. Um dies zu beheben, verwenden wir die "Vertikal"-Steuerungen in der Box auf der rechten Seite des Bildschirms. Diese Regler wirken sich auf den aktuell ausgewählten Kanal aus. Bei der Einstellung der Verstärkung ist zu beachten, dass sie einen riesigen Bereich abdeckt - im Gegensatz zu einem echten Oszilloskop kann dieses Gerät Signale von sehr kleinen (Pico-Einheiten) bis zu sehr großen (Tera-Einheiten) anzeigen. Mit dem Positionsregler wird die angezeigte Kurve nur über die Höhe des Bildschirms nach oben und unten bewegt. Für größere Einstellungen sollte die Offset-Taste verwendet werden.

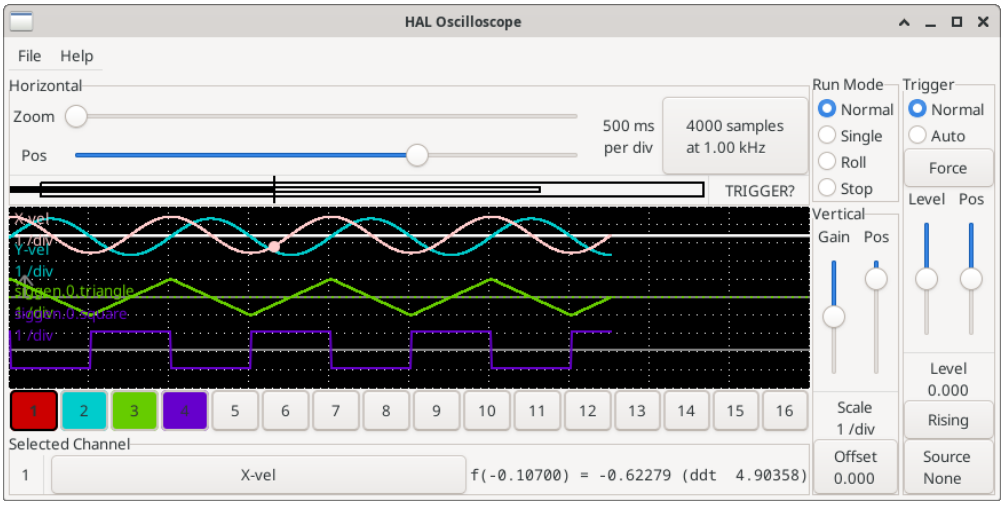


Figure 87. Vertikale Einstellung

Die große Schaltfläche *Ausgewählter Kanal* am unteren Rand zeigt an, dass Kanal 1 der aktuell ausgewählte Kanal ist und dass er mit dem *X-vel*-Signal übereinstimmt. Versuchen Sie, auf die anderen Kanäle zu klicken, um ihre Spuren sichtbar zu machen und sie mit dem *Pos*-Cursor verschieben zu können.

Triggering (automatisches Auslösen)

Die Verwendung des Button "Erzwingen" ist eine eher unbefriedigende Art, das Oszilloskop auszulösen. Um eine echte Triggerung einzurichten, klicken Sie auf die Schaltfläche "Quelle" unten rechts. Daraufhin wird das Dialogfeld "Trigger Source" (Triggerquelle) angezeigt, das einfach eine Liste aller derzeit angeschlossenen Sonden enthält. Wählen Sie eine Sonde für die Triggerung aus, indem Sie auf sie klicken. In diesem Beispiel verwenden wir Kanal 3, die Dreieckswelle, wie in der folgenden Abbildung dargestellt.

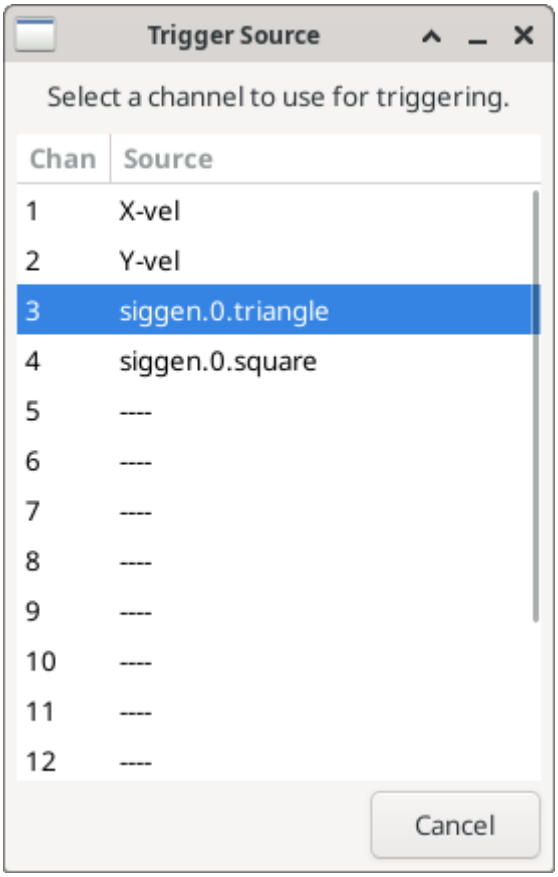


Figure 88. Dialogfeld Triggerquelle (engl. trigger source)

Nachdem Sie die Triggerquelle eingestellt haben, können Sie den Triggerpegel und die Triggerposition mit den Schiebereglern im Feld "Trigger" am rechten Rand einstellen. Der Pegel kann vom oberen bis zum unteren Rand des Bildschirms eingestellt werden und wird unter den Schiebereglern angezeigt. Die Position ist die Lage des Auslösepunkts innerhalb der gesamten Aufzeichnung. Ist der Schieberegler ganz unten, befindet sich der Auslösepunkt am Ende der Aufzeichnung, und halscope zeigt an, was vor dem Auslösepunkt passiert ist. Wenn der Schieberegler ganz nach oben geschoben ist, befindet sich der Auslösepunkt am Anfang des Datensatzes und es wird angezeigt, was nach dem Auslösen passiert ist. Der Triggerpunkt ist als vertikale Linie in der Fortschrittsanzeige über dem Bildschirm sichtbar. Die Triggerpolarität kann durch Klicken auf die Schaltfläche direkt unter der Triggerpegelanzeige geändert werden. Sie wird dann *absteigend*. Beachten Sie, dass die Änderung der Triggerposition das Oszilloskop anhält, sobald die Position angepasst wurde, starten Sie das Oszilloskop erneut, indem Sie auf die Schaltfläche *Normal* des *Run-Modus* der Gruppe klicken.

Nachdem wir nun die vertikalen Regler und die Triggerung eingestellt haben, sieht die Anzeige des Oszilloskops etwa wie in der folgenden Abbildung aus.

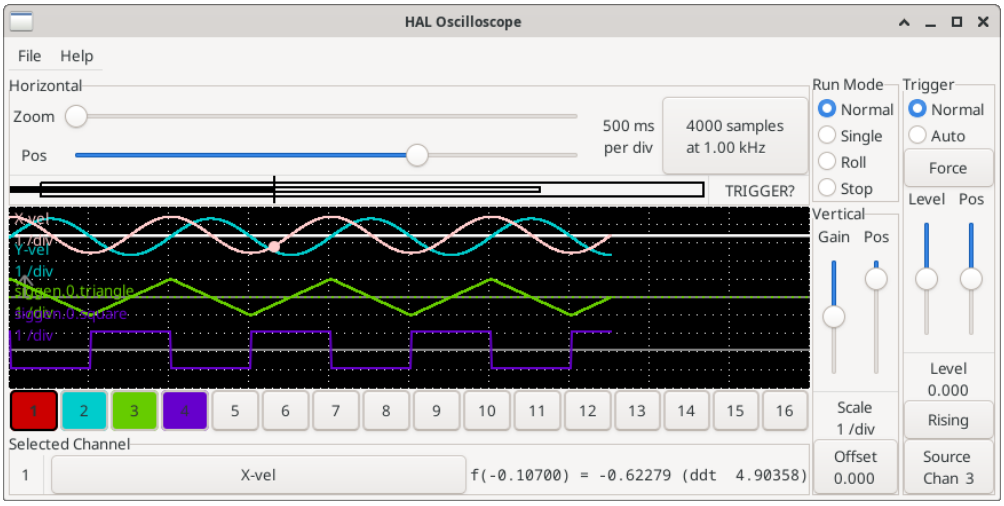


Figure 89. Wellenformen mit Triggerung

Horizontale Anpassungen

Um einen Teil einer Wellenform genauer zu betrachten, können Sie den Zoom-Schieberegler am oberen Rand des Bildschirms verwenden, um die Wellenformen horizontal zu erweitern, und den Positionsschieberegler, um zu bestimmen, welcher Teil der gezoomten Wellenform sichtbar ist. Manchmal reicht es jedoch nicht aus, die Wellenformen einfach zu vergrößern, und Sie müssen die Abtastrate erhöhen. Wir möchten uns zum Beispiel die tatsächlichen Schrittimpulse ansehen, die in unserem Beispiel erzeugt werden. Da die Schrittimpulse möglicherweise nur 50 µs lang sind, ist eine Abtastrate von 1 kHz nicht schnell genug. Um die Abtastrate zu ändern, klicken Sie auf die Schaltfläche, welche die Anzahl der Abtastungen und die Abtastrate anzeigt, um das Dialogfeld "Abtastrate auswählen" aufzurufen, Abbildung . In diesem Beispiel klicken wir auf den 50 µs-Thread "schnell", wodurch wir eine Abtastrate von etwa 20 kHz erhalten. Statt 4 Sekunden Daten anzuzeigen, besteht ein Datensatz nun aus 4000 Samples bei 20 kHz, also etwa 0,20 Sekunden.

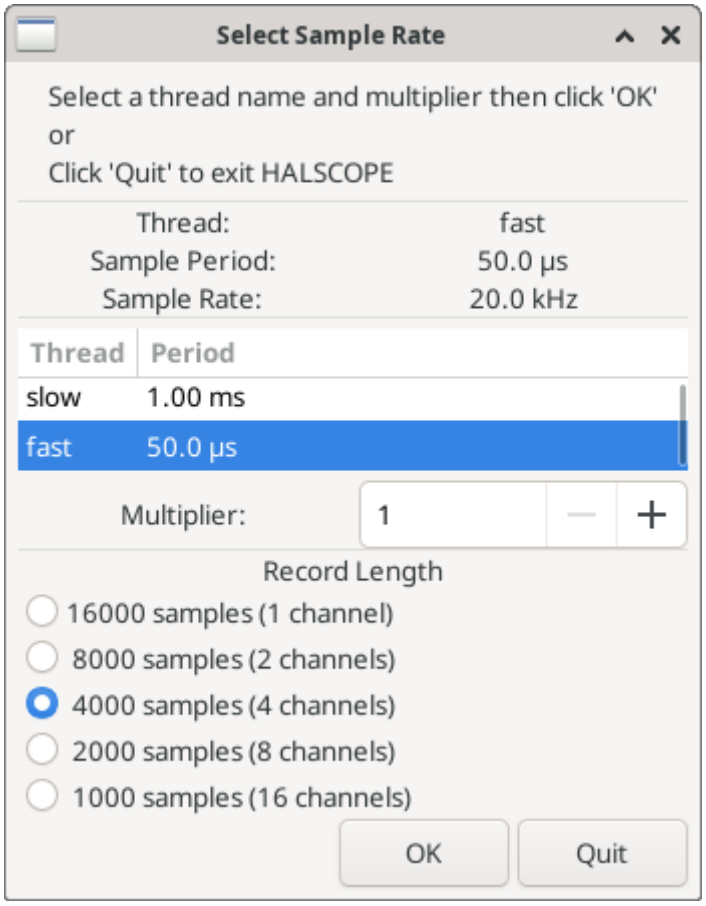


Figure 90. Dialogfeld für Abtastrate

Weitere Kanäle

Schauen wir uns nun die Schrittimpulse an. Halscope hat 16 Kanäle, aber in diesem Beispiel verwenden wir jeweils nur 4. Bevor wir weitere Kanäle auswählen, müssen wir ein paar ausschalten. Ein Klick auf eine ausgewählte Kanal-Schaltfläche (schwarzer Rahmen) schaltet den Kanal aus. Klicken Sie also auf die Schaltfläche für Kanal 2, klicken Sie dann noch einmal darauf und der Kanal wird ausgeschaltet. Danach klicken Sie zweimal auf Kanal 3 und machen dasselbe für Kanal 4. Obwohl die Kanäle ausgeschaltet sind, „merken“ sie sich weiterhin, womit sie verbunden sind – und tatsächlich werden wir Kanal 3 weiterhin als Triggerquelle nutzen. Um neue Kanäle hinzuzufügen, wählen Sie Kanal 5 und ordnen diesem den Pin `stepgen.0.dir` zu, anschließend Kanal 6 mit `stepgen.0.step`. Starten Sie dann mit „Run Mode: Normal“ das Oszilloskop und stellen Sie den horizontalen Zoom auf 5 ms pro Teilung ein. Sie sollten sehen, wie die Schrittimpulse langsamer werden, wenn das Geschwindigkeitskommando (Kanal 1) gegen Null geht. Dann ändert sich der Status des Richtungspins, und die Schrittimpulse werden wieder schneller. Möglicherweise sollten Sie die Verstärkung von Kanal 1 auf etwa 20 Milli pro Teilung erhöhen, um die Änderung des Geschwindigkeitskommandos besser zu erkennen. Das Ergebnis sollte wie in der folgenden Abbildung aussehen.

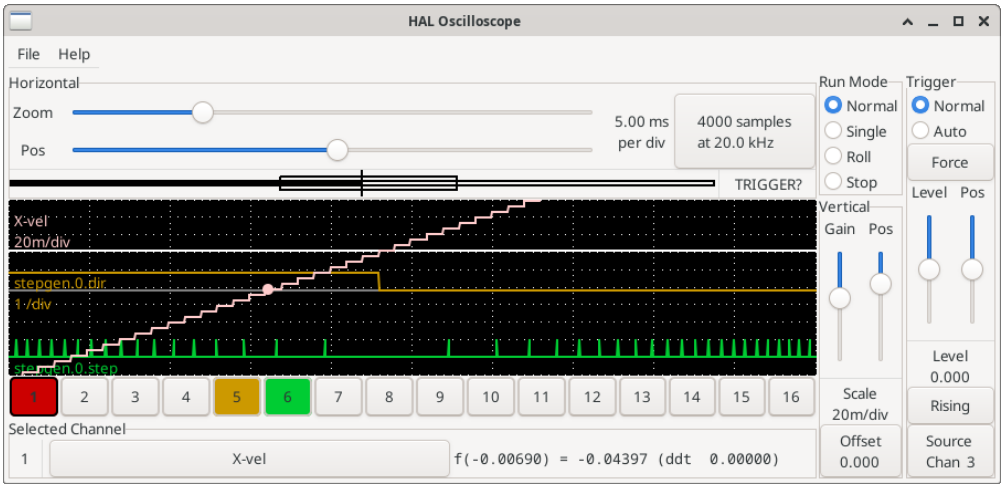


Figure 91. Schrittimpulse

Weitere Samples

Wenn Sie mehr Samples auf einmal aufnehmen wollen, starten Sie realtime neu und laden Sie halscope mit einem numerischen Argument, das die Anzahl der Samples angibt, die Sie aufnehmen wollen.

```
halcmd loadusr halscope 80000
```

Wenn die Komponente *scope_rt* noch nicht geladen war, lädt halscope sie und fordert 80000 Gesamtsamples an, so dass bei der Abtastung von 4 Kanälen gleichzeitig 20000 Samples pro Kanal zur Verfügung stehen. (Wenn *scope_rt* bereits geladen war, hat das numerische Argument für halscope keine Auswirkungen).

5.6. HAL Beispiele

Bei all diesen Beispielen wird davon ausgegangen, dass Sie mit einer stepconf-basierten Konfiguration beginnen und zwei Threads *base-thread* und *servo-thread* haben. Der StepConf-Assistent erstellt eine leere Datei custom.hal und eine Datei custom_postgui.hal. Die Datei custom.hal wird nach der Konfigurations-HAL-Datei geladen und die Datei custom_postgui.hal wird geladen, nachdem die GUI geladen wurde.

5.6.1. Verbinden von zwei Ausgängen

Um zwei Ausgänge mit einem Eingang zu verbinden, können Sie die Komponente or2 verwenden. Die or2-Komponente funktioniert folgendermaßen: Wenn einer der beiden Eingänge von or2 eingeschaltet ist, dann ist der or2-Ausgang eingeschaltet. Wenn keiner der beiden Eingänge von or2 eingeschaltet ist, dann ist der or2-Ausgang ausgeschaltet.

Zum Beispiel, um zwei PyVCP-Tasten zu haben, die beide an eine LED angeschlossen sind.

Die .xml-Datei, die PyVCP anweist, eine grafische Benutzeroberfläche mit zwei Schaltflächen (namens "button-1" und "button-2") und einer LED (namens "led-1") vorzubereiten.

```
<pyvcp>
  <button>
    <halpin>"button-1"</halpin>
    <text>"Button 1"</text>
  </button>

  <button>
    <halpin>"button-2"</halpin>
    <text>"Button 2"</text>
  </button>

  <led>
    <halpin>"led-1"</halpin>
    <size>50</size>
    <on_color>"green"</on_color>
    <off_color>"red"</off_color>
  </led>
</pyvcp>
```

Die Datei postgui.hal, die gelesen wird, nachdem die grafische Benutzeroberfläche eingerichtet wurde und die Ports bereit sind, die in HAL beschriebene Logik zu übernehmen.

```
loadrt or2
addf or2.0 servo-thread
net button-1 or2.0.in0 <= pyvcp.button-1
net button-2 or2.0.in1 <= pyvcp.button-2
net led-1 pyvcp.led-1 <= or2.0.out
```

Wenn Sie dieses Beispiel in einem mit dem StepConf-Assistenten erstellten Achsen-Simulator ausführen, können Sie ein Terminal öffnen und die mit *loadrt or2* erstellten Pins sehen, indem Sie `halcmd show pin or2` in das Terminal eingeben.

Ausführung von `halcmd` *in der UNIX-Befehlszeile, um die mit dem Modul or2 bereitgestellten Pins anzuzeigen.*

```
$ halcmd show pin or2
Component Pins:
Owner   Type  Dir         Value  Name
    22  bool  IN          FALSE  or2.0.in0 <== button-1
    22  bool  IN          FALSE  or2.0.in1 <== button-2
    22  bool  OUT         FALSE  or2.0.out ==> led-1
```

Aus dem HAL-Befehl `show pin or2` geht hervor, dass der Pin `button-1` mit dem Pin `or2.0.in0` verbunden ist, und aus dem Richtungspfeil geht hervor, dass der Button ein Ausgang und der `or2.0.in0` ein Eingang ist. Der Ausgang von or2 geht an den Eingang der LED.

5.6.2. Manueller Werkzeugwechsel

In diesem Beispiel wird davon ausgegangen, dass Sie Ihre eigene Konfiguration erstellen und das Fenster HAL Manual Toolchange hinzufügen möchten. Der manuelle HAL-Werkzeugwechsel ist vor

allem dann nützlich, wenn Sie voreinstellbare Werkzeuge haben und die Offsets in der Werkzeugtabelle speichern. Wenn Sie für jeden Werkzeugwechsel einen neuen Wert eingeben müssen, ist es am besten, wenn Sie Ihren G-Code aufteilen. Um das HAL Manual Toolchange-Fenster zu verwenden, müssen Sie grundsätzlich

1. die hal_manualtoolchange Komponente laden,
2. dann das iocontrol "tool change" an das hal_manualtoolchange "change" senden und
3. das hal_manualtoolchange *changed* zurück an das iocontrol *tool changed* senden.

Ein Pin ist für einen externen Eingang vorgesehen, der anzeigt, dass der Werkzeugwechsel abgeschlossen ist.

Dies ist ein Beispiel für den manuellen Werkzeugwechsel *mit* der Komponente HAL Manual Toolchange:

```
loadusr -W hal_manualtoolchange
net tool-change iocontrol.0.tool-change => hal_manualtoolchange.change
net tool-changed iocontrol.0.tool-changed <= hal_manualtoolchange.changed
net external-tool-changed hal_manualtoolchange.change_button <= parport.0.pin-12-in
net tool-number iocontrol.0.tool-prep-number => hal_manualtoolchange.number
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

Dies ist ein Beispiel für den manuellen Werkzeugwechsel *ohne* die Komponente HAL Manual Toolchange:

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

5.6.3. Geschwindigkeit berechnen

Dieses Beispiel verwendet *ddt*, *mult2* und *abs*, um die Geschwindigkeit einer einzelnen Achse zu berechnen. Weitere Informationen zu den Echtzeitkomponenten finden Sie in den Man Pages oder in der HAL Components List (HAL-Komponenten).

Als Erstes sollten Sie Ihre Konfiguration überprüfen, um sicherzustellen, dass Sie keine der Echtzeitkomponenten bereits verwenden. Öffnen Sie dazu das HAL-Konfigurationsfenster und suchen Sie nach den Komponenten im Pin-Bereich. Wenn dies der Fall ist, suchen Sie die .hal-Datei, in die sie geladen werden, und erhöhen Sie die Anzahl und passen Sie die Instanz auf den richtigen Wert an. Fügen Sie Folgendes zu Ihrer custom.hal-Datei hinzu.

Laden der Echtzeitkomponenten.

```
loadrt ddt count=1
loadrt mult2 count=1
loadrt abs count=1
```

Fügen Sie die Funktionen in einen Thread ein, damit dieser aktualisiert wird.

```
addf ddt.0 servo-thread
addf mult2.0 servo-thread
addf abs.0 servo-thread
```

Herstellen der Verbindungen.

```
setp mult2.in1 60
net xpos-cmd ddt.0.in
net X-IPS mult2.0.in0 <= ddt.0.out
net X-ABS abs.0.in <= mult2.0.out
net X-IPM abs.0.out
```

In diesem letzten Abschnitt setzen wir `mult2.0.in1` auf 60, um die Zoll pro Sekunde in Zoll pro Minute (engl. IPM für inch per minute) umzuwandeln, die wir von `ddt.0.out` erhalten.

Der `xpos-cmd` sendet die befohlene Position an den `ddt.0.in`. Der ddt berechnet die Ableitung der Änderung des Eingangs.

Der `ddt2.0.out` wird mit 60 multipliziert und ergibt die IPM (Zoll/min).

Der `mult2.0.out` wird an den `abs` gesendet, um den absoluten Wert zu erhalten.

Die folgende Abbildung zeigt das Ergebnis, wenn sich die X-Achse mit 15 IPM in die Minusrichtung bewegt. Beachten Sie, dass wir den absoluten Wert entweder vom `abs.0.out`-Pin oder dem `X-IPM` -Signal erhalten können.

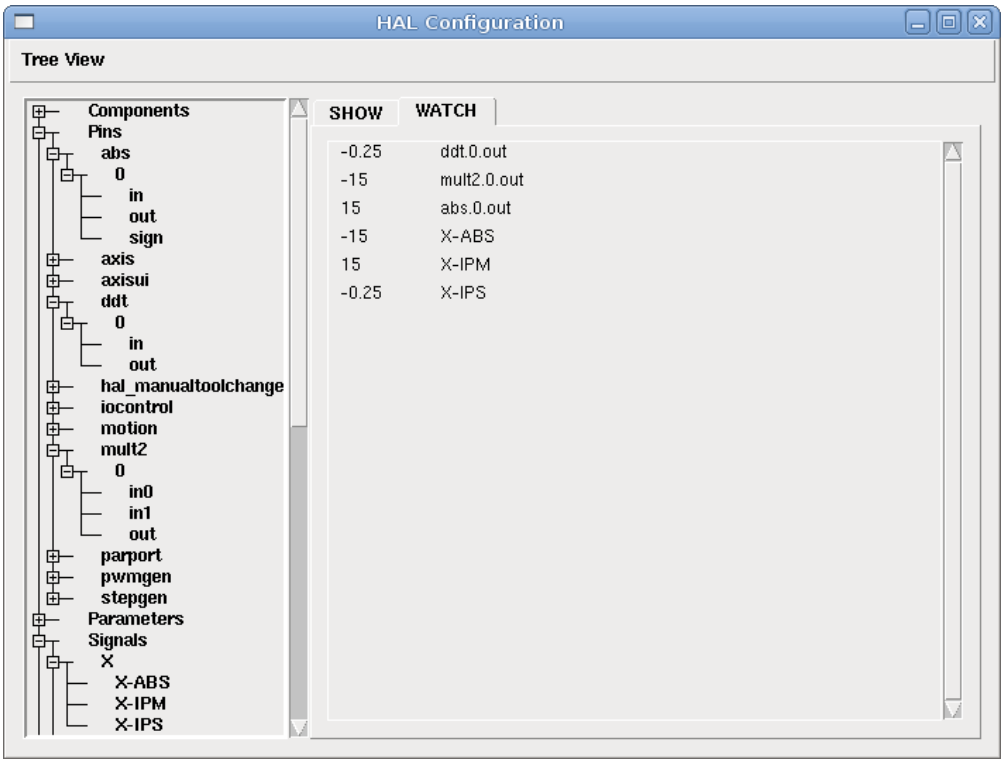


Figure 92. HAL: Geschwindigkeitsbeispiel

5.6.4. Details zum Softstart

Dieses Beispiel zeigt, wie die HAL-Komponenten *Tiefpass*, *limit2* oder *limit3* verwendet werden können, um die Änderungsgeschwindigkeit eines Signals zu begrenzen.

In diesem Beispiel haben wir einen Servomotor, der eine Drehbankspindel antreibt. Wenn wir nur die befohlenen Spindeldrehzahlen für den Servomotor verwenden würden, würde er versuchen, so schnell wie möglich von der aktuellen Drehzahl auf die befohlene Drehzahl zu kommen. Dies könnte zu Problemen führen oder den Antrieb beschädigen. Um die Änderungsrate zu verlangsamen, können wir die `spindle.N.speed-out` durch einen Begrenzer vor dem PID senden, so dass sich der PID-Befehlswert langsamer auf neue Einstellungen ändert.

Die drei eingebauten Komponenten, die ein Signal begrenzen, sind:

- *limit2* begrenzt den Bereich und die erste Ableitung eines Signals.
- *limit3* begrenzt den Bereich, die erste und zweite Ableitung eines Signals.
- *Tiefpass* verwendet einen exponentiell gewichteten gleitenden Durchschnitt, um ein Eingangssignal zu verfolgen.

Weitere Informationen über diese HAL-Komponenten finden Sie in den Man Pages.

Geben Sie das Folgende in eine Textdatei namens softstart.hal ein. Wenn Sie mit Linux nicht vertraut sind, legen Sie die Datei in Ihr Home-Verzeichnis.

```
loadrt threads period1=1000000 name1=thread
loadrt siggen
loadrt lowpass
loadrt limit2
loadrt limit3
net square siggen.0.square => lowpass.0.in limit2.0.in limit3.0.in
net lowpass <= lowpass.0.out
net limit2 <= limit2.0.out
net limit3 <= limit3.0.out
setp siggen.0.frequency .1
setp lowpass.0.gain .01
setp limit2.0.maxv 2
setp limit3.0.maxv 2
setp limit3.0.maxa 10
addf siggen.0.update thread
addf lowpass.0 thread
addf limit2.0 thread
addf limit3.0 thread
start
loadusr halscope
```

Öffnen Sie ein Terminalfenster und führen Sie die Datei mit dem folgenden Befehl aus.

```
halrun -I softstart.hal
```

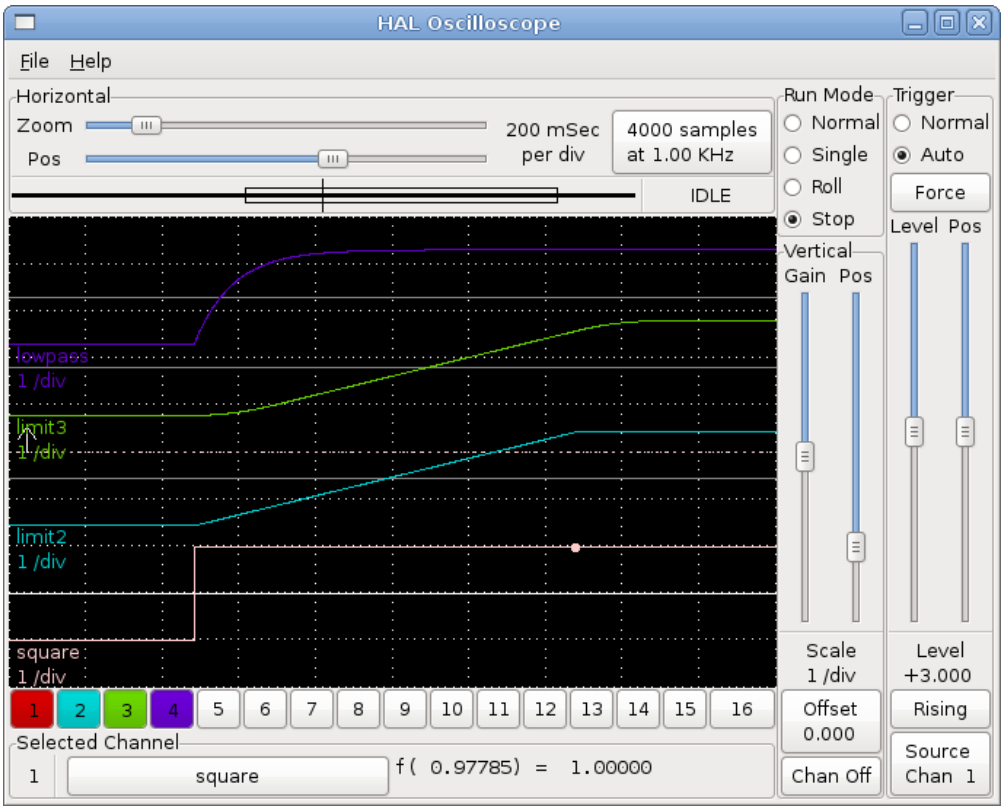
Wenn das HAL Oszilloskop zum ersten Mal startet, klicken Sie auf *OK*, um den Standardfaden zu akzeptieren.

Als nächstes müssen Sie die Signale zu den Kanälen hinzufügen. Klicken Sie auf Kanal 1 und wählen Sie dann *Quadrat* auf der Registerkarte Signale. Wiederholen Sie dies für die Kanäle 2-4 und fügen Sie Tiefpass, Limit2 und Limit3 hinzu.

Um ein Triggersignal einzurichten, klicken Sie auf die Schaltfläche Source None und wählen Sie Square. Die Schaltfläche ändert sich in Quelle Kanal 1.

Als nächstes klicken Sie im Optionsfeld Run Mode auf Single. Dadurch wird ein Lauf gestartet, und nach dessen Beendigung sehen Sie die Spuren (engl. traces).

Um die Signale zu trennen, damit Sie sie besser sehen können, klicken Sie auf einen Kanal und verwenden Sie dann den Pos-Schieberegler im vertikalen Feld, um die Positionen festzulegen.



Um zu sehen, wie sich eine Änderung der Sollwerte der einzelnen Komponenten auswirkt, können Sie diese im Terminalfenster ändern. Um zu sehen, was verschiedene Verstärkungseinstellungen für den Tiefpass bewirken, geben Sie einfach Folgendes in das Terminalfenster ein und probieren Sie verschiedene Einstellungen aus.

```
setp lowpass.0.gain *.01
```

Nachdem Sie eine Einstellung geändert haben, lassen Sie das Oszilloskop erneut laufen, um die Änderung zu sehen.

Wenn Sie fertig sind, geben Sie *exit* in das Terminalfenster ein, um halrun zu beenden und das halscope zu schließen. Schließen Sie das Terminal-Fenster nicht, während halrun läuft, da es einige Dinge im Speicher lassen könnte, die das Laden von LinuxCNC verhindern könnten.

Weitere Informationen zu Halscope finden Sie im HAL-Handbuch und im Tutorial.

5.6.5. Stand-Alone HAL

In manchen Fällen möchten Sie vielleicht einen GladeVCP-Bildschirm nur mit HAL betreiben. Nehmen wir an, Sie haben ein schrittmotorgesteuertes Gerät und alles was Sie benötigen ist diesen Schrittmotor zusteuern. Eine einfache *Start/Stop* Schnittstelle ist alles, was Sie für Ihre Anwendung benötigen, so dass Sie keine vollständige CNC-Anwendung laden und konfigurieren müssen.

Im folgenden Beispiel haben wir ein einfaches GladeVCP-Panel mit einem Schrittmotor.

Grundlegende (engl. basic) Syntax

```
# Lädt die GUI winder.glade und nennt diese winder
loadusr -Wn winder gladevcp -c winder -u handler.py winder.glade

# load Echtzeit-Komponenten
loadrt threads name1=fast period1=50000 fp1=0 name2=slow period2=1000000
loadrt stepgen step_type=0 ctrl_type=v
loadrt hal_parport cfg="0x378 out"

# fügt Funktionen den Threads hinzu
addf stepgen.make-pulses fast
addf stepgen.update-freq slow
addf stepgen.capture-position slow
addf parport.0.read fast
addf parport.0.write fast

# Erstellt Verbindungen in HAL
net winder-step parport.0.pin-02-out <= stepgen.0.step
net winder-dir parport.0.pin-03-out <= stepgen.0.dir
net run-stepgen stepgen.0.enable <= winder.start_button

# Start der Threads
start

# kommentieren Sie die folgenden Zeilen beim Testen aus und verwenden Sie die interaktive
# option halrun -I -f start.hal, um Pins etc. anzeigen zu können.

# Warten, bis die GladeVCP-GUI namens winder beendet ist
waitusr winder

# Stop HAL threads
stop

# "unload" aller Komponenten in HAL vor dem Beenden
unloadrt all
```

5.7. Kern-Komponenten

Siehe auch die Man Pages zu *motion(9)*.

5.7.1. Bewegung (engl. motion)

Diese Pins und Parameter werden durch das Echtzeitmodul *motmod* erzeugt.

Dieses Modul bietet eine HAL-Schnittstelle für den Bewegungsplaner (engl. motion planner) von LinuxCNC.

Im Grunde genommen nimmt motmod eine Liste von Wegpunkten auf und erzeugt daraus einen schönen ineinander übergehenden und durch Einschränkungen begrenzten Strom von Gelenkpositionen, der an die Motorantriebe weitergeleitet wird.

Optional wird die Anzahl der digitalen E/A (engl. I/O) mit *num_dio* eingestellt. Die Anzahl der analogen E/A wird mit *num_aio* festgelegt, Standard ist jeweils 4. Die Anzahl der Spindeln wird mit *num_spindles* eingestellt, Voreinstellung ist 1.

Pin- und Parameternamen, die mit *axis.L* und *joint.N* beginnen, werden von der Motion-Controller-Funktion gelesen und aktualisiert.

Motion wird mit dem Befehl motmod geladen. Ein kins sollte vor motion geladen werden.

```
loadrt motmod base_period_nsec=['period'] servo_period_nsec=['period']
             traj_period_nsec=['period'] num_joints=['0-9']
             num_dio=['1-64'] num_aio=['1-16'] unlock_joints_mask=['0xNN']
             num_spindles=['1-8']
```

- *base_period_nsec = 50000* - die *Basis*-Task Period in Nanosekunden. Dies ist der schnellste Thread der Maschine.

NOTE Bei Servo-basierten Systemen gibt es im Allgemeinen keinen Grund dafür, dass *base_period_nsec* kleiner ist als *servo_period_nsec*. Bei Maschinen mit Software-Schrittgenerierung bestimmt die *base_period_nsec* die maximale Anzahl der Schritte pro Sekunde. Wenn keine großen Schrittlängen und Schrittabstände erforderlich sind, beträgt die absolut maximale Schrittrate einen Schritt pro *base_period_nsec*. Somit ergibt die oben gezeigte *base_period_nsec* eine absolute maximale Schrittrate von 20.000 Schritten pro Sekunde. 50.000 ns (50 µs) ist ein recht konservativer Wert. Der kleinste brauchbare Wert hängt mit dem Ergebnis des Latenztests, der erforderlichen Schrittlänge und der Prozessorgeschwindigkeit zusammen. Die Wahl einer zu niedrigen *base_period_nsec* kann zu der Meldung "Unerwartete Echtzeitverzögerung", zu Blockierungen oder spontanen Reboots führen.

- *servo_period_nsec = 1000000* - Dies ist die *Servo* task period in Nanosekunden. Dieser Wert wird auf ein ganzzahliges Vielfaches von *base_period_nsec* gerundet. Diese Periode wird auch bei Systemen verwendet, die auf Schrittmotoren basieren.

  Dies ist die Rate, mit der neue Motorpositionen berechnet werden, Schleppfehler überprüft werden, PID-Ausgangswerte aktualisiert werden und so weiter. Die meisten Systeme werden diesen Wert nicht ändern müssen. Es ist die Aktualisierungsrate des Low-Level-Bewegungsplaners.

- *traj_period_nsec = 100000* - Dies ist die *Trajectory Planner* Aufgabenperiode in Nanosekunden. Dieser Wert wird auf ein ganzzahliges Vielfaches von *servo_period_nsec* gerundet. Außer bei Maschinen mit ungewöhnlicher Kinematik (z.B. Hexapods) gibt es keinen Grund, diesen Wert größer als *servo_period_nsec* zu machen.

Optionen

Wenn Sie mehr als die standardmäßige Anzahl von 4 digitalen E/A benötigen, können Sie bis zu 64 digitale E/A hinzufügen, indem Sie die Option num_dio beim Laden von *motmod* verwenden.

Wenn mehr als die voreingestellte Anzahl von 4 analogen E/A benötigt wird, können Sie bis zu 16 analoge E/A hinzufügen, indem Sie die Option num_aio beim Laden von *motmod* verwenden.

Der Parameter unlock_joints_mask wird verwendet, um Pins für ein Gelenk zu erzeugen, das als verriegelnder Indexer verwendet wird (normalerweise ein Drehgelenk). Die Maskenbits wählen das/die Gelenk(e) aus. Das LSB der Maske wählt das Gelenk 0. Beispiel:

```
unlock_joints_mask=0x38 wählt die Gelenke 3,4,5 aus
```

Pins

Diese Pins, Parameter und Funktionen werden durch das Echtzeitmodul *motmod* angelegt.

- *motion-command-handler.time* - (real, out) runtime in seconds
- *motion-command-handler.tmax* - (real, io) max runtime in seconds, write 0 to reset
- *motion-controller.time* - (real, out) runtime in seconds
- *motion-controller.tmax* - (real, io) max runtime in seconds, write 0 to reset
- *motion.adaptive-feed* - (real, in) When adaptive feed is enabled with *M52 P1* , the commanded velocity is multiplied by this value. This effect is multiplicative with the NML-level feed override value and *motion.feed-hold*. As of version 2.9 of LinuxCNC it is possible to use a negative adaptive feed value to run the G-code path in reverse.
- *motion.analog-in-00* - (real, in) These pins (00, 01, 02, 03 or more if configured) are controlled by M66.
- *motion.analog-out-00* - (real, out) These pins (00, 01, 02, 03 or more if configured) are controlled by M67 or M68.
- *motion.coord-error* - (bool, out) TRUE when motion has encountered an error, such as exceeding a soft limit
- *motion.coord-mode* - (bool, out) TRUE when motion is in *coordinated mode*, as opposed to *teleop mode*
- *motion.current-vel* - (real, out) The current tool velocity in user units per second.
- *motion.digital-in-00* - (bool, in) These pins (00, 01, 02, 03 or more if configured) are controlled by M62-65.
- *motion.digital-out-00* - (bool, out) These pins (00, 01, 02, 03 or more if configured) are controlled by the *M62-65*.
- *motion.distance-to-go* - (real, out) The distance remaining in the current move.
- *motion.enable* - (bool, in) If this input is driven FALSE, motion stops, the machine is placed in the *machine off* state, and a message is displayed for the operator. For normal motion, drive this input TRUE.
- *motion.feed-hold* - (bool, in) When Feed Stop Control is enabled with *M53 P1*, and this input is TRUE,

the feed rate is set to 0.

- *motion.feed-inhibit* - (bool, in) When this input is TRUE, the feed rate is set to 0. This will be delayed during spindle synch moves till the end of the move.
- *motion.in-position* - (bool, out) TRUE if the machine is in position.
- *motion.motion-enabled* - (bool, out) TRUE when in *machine on* state.
- *motion.motion-type* - (sint, out) These values are from include/motion_types.h
  - 0: Leerlauf (engl. "idle", d.h. keine Bewegung)
  - 1: Traverse (direkte Bewegung)
  - 2: Linearer Vorschub
  - 3: Kreisbogenvorschub
  - 4: Werkzeugwechsel
  - 5: Antasten
  - 6: Indexierung der Drehachse
- *motion.on-soft-limit* - (bool, out) TRUE when the machine is on a soft limit.
- *motion.probe-input* - (bool, in) *G38.n* uses the value on this pin to determine when the probe has made contact. TRUE for probe contact closed (touching), FALSE for probe contact open.
- *motion.program-line* - (sint, out) The current program line while executing. Zero if not running or between lines while single stepping.
- *motion.requested-vel* - (real, out) The current requested velocity in user units per second. This value is the F-word setting from the G-code file, possibly reduced to accommodate machine velocity and acceleration limits. The value on this pin does not reflect the feed override or any other adjustments.
- *motion.teleop-mode* - (bool, out) TRUE when motion is in *teleop mode*, as opposed to *coordinated mode*
- *motion.tooloffset.x ... motion.tooloffset.w* - (real, out, one per axis) shows the tool offset in effect; it could come from the tool table (*G43* active), or it could come from the G-code (*G43.1* active)
- *motion.on-soft-limit* - (bool, out) TRUE when the machine is on a soft limit.
- *motion.probe-input* - (bool, in) *G38.n* uses the value on this pin to determine when the probe has made contact. TRUE for probe contact closed (touching), FALSE for probe contact open.
- *motion.program-line* - (sint, out) The current program line while executing. Zero if not running or between lines while single stepping.
- *motion.requested-vel* - (real, out) The current requested velocity in user units per second. This value is the F-word setting from the G-code file, possibly reduced to accommodate machine velocity and acceleration limits. The value on this pin does not reflect the feed override or any other adjustments.
- *motion.teleop-mode* - (bool, out) TRUE when motion is in *teleop mode*, as opposed to *coordinated mode*
- *motion.tooloffset.x ... motion.tooloffset.w* - (real, out, one per axis) shows the tool offset in effect; it could come from the tool table (*G43* active), or it could come from the G-code (*G43.1* active)

Parameter

Viele dieser Parameter dienen als Hilfsmittel zur Fehlersuche und können jederzeit geändert oder entfernt werden.

- *motion.debug-bool-0* - (bool, RO) This is used for debugging purposes.
- *motion.debug-bool-1* - (bool, RO) This is used for debugging purposes.
- *motion.debug-real-0* - (real, RO) This is used for debugging purposes.
- *motion.debug-real-1* - (real, RO) This is used for debugging purposes.
- *motion.debug-real-2* - (real, RO) This is used for debugging purposes.
- *motion.debug-real-3* - (real, RO) This is used for debugging purposes.
- *motion.debug-sint-0* - (sint, RO) This is used for debugging purposes.
- *motion.debug-sint-1* - (sint, RO) This is used for debugging purposes.
- *motion.servo.last-period* - (uint, RO) The time in ns between invocations of the servo thread. This number can be used to determine whether the realtime motion controller is meeting its timing constraints

Funktionen

Im Allgemeinen werden diese beiden Funktionen in der angegebenen Reihenfolge zum Servo-Thread hinzugefügt.

- *motion-command-handler* - Empfängt und verarbeitet Bewegungsbefehle aus dem Benutzerbereich
- *motion-controller* - Führt die LinuxCNC Bewegungssteuerung aus

5.7.2. Spindel

LinuxCNC kann bis zu acht Spindeln steuern. Motion wird die folgenden Pins anlegen: Das *N* (ganze Zahl zwischen 0 und 7) entspricht der Spindel-Nummer.

Pins

- *spindle.N.at-speed* - (bool, in) Motion will pause until this pin is TRUE, under the following conditions:
  - vor der ersten Vorschubbewegung nach jedem Spindelstart oder Drehzahlwechsel;
  - vor dem Beginn jeder Kette von spindelsynchronisierten Bewegungen;
  - und im CSS-Modus bei jedem Übergang von Eilgang zu Vorschub. Dieser Eingang kann verwendet werden, um sicherzustellen, dass die Spindel vor dem Beginn eines Schnittes die volle Drehzahl erreicht hat oder dass eine Drehmaschinenspindel im CSS-Modus nach einem Durchgang vom großen zum kleinen Plandurchmesser abgebremst hat, bevor der nächste Durchgang am großen Durchmesser beginnt. Viele VFDs haben einen *bei Drehzahl* Ausgang. Andernfalls ist es einfach, dieses Signal mit der Komponente *HAL near* zu erzeugen, indem man

die gewünschte und die tatsächliche Spindeldrehzahl vergleicht.

- *spindle.N.brake* - (bool, out) TRUE when the spindle brake should be applied.
- *spindle.N.forward* - (bool, out) TRUE when the spindle should rotate forward.
- *spindle.N.index-enable* - (bool, I/O) For correct operation of spindle synchronized moves, this pin must be hooked to the index-enable pin of the spindle encoder.
- *spindle.N.inhibit* - (bool, in) When this input is TRUE, the spindle speed is set to 0.
- *spindle.N.on* - (bool, out) TRUE when spindle should rotate.
- *spindle.N.reverse* - (bool, out) TRUE when the spindle should rotate backward
- *spindle.N.revs* - (real, in) For correct operation of spindle synchronized moves, this signal must be hooked to the position pin of the spindle encoder. The spindle encoder position should be scaled such that spindle-revs increases by 1.0 for each rotation of the spindle in the clockwise (*M3*) direction.
- *spindle.N.speed-in* - (real, in) Feedback of actual spindle speed in rotations per second. This is used by feed-per-revolution motion (*G95*). If your spindle encoder driver does not have a velocity output, you can generate a suitable one by sending the spindle position through a *ddt* component. If you do not have a spindle encoder, you can loop back *spindle.N.speed-out-rps*.
- *spindle.N.speed-out* - (real, out) Commanded spindle speed in rotations per minute. Positive for spindle forward (*M3*), negative for spindle reverse (*M4*).
- *spindle.N.speed-out-abs* - (real, out) Commanded spindle speed in rotations per minute. This will always be a positive number.
- *spindle.N.speed-out-rps* - (real, out) Commanded spindle speed in rotations per second. Positive for spindle forward (*M3*), negative for spindle reverse (*M4*).
- *spindle.N.speed-out-rps-abs* - (real, out) Commanded spindle speed in rotations per second. This will always be a positive number.
- *spindle.N.orient-angle* - (real, out) Desired spindle orientation for M19. Value of the M19 R word parameter plus the value of the [RS274NGC]ORIENT_OFFSET INI parameter.
- *spindle.N.orient-mode* - (sint, out) Desired spindle rotation mode M19. Default 0.
- *spindle.N.orient* - (bool, out) Indicates start of spindle orient cycle. Set by M19. Cleared by any of M3, M4, or M5. If spindle-orient-fault is not zero during spindle-orient true, the M19 command fails with an error message.
- *spindle.N.is-oriented* - (bool, in) Acknowledge pin for spindle-orient. Completes orient cycle. If spindle-orient was true when spindle-is-oriented was asserted, the spindle-orient pin is cleared and the spindle-locked pin is asserted. Also, the spindle-brake pin is asserted.
- *spindle.N.orient-fault* - (sint, in) Fault code input for orient cycle. Any value other than zero will cause the orient cycle to abort.
- *spindle.N.lock* - (bool, out) Spindle orient complete pin. Cleared by any of M3, M4, or M5.

Verwendung des HAL-Pins für die M19-Spindel-Orientierung

Konzeptionell befindet sich die Spindel in einem der folgenden Modi:

- Rotationsmodus (die Standardeinstellung)
- Suchend nach der gewünschten Orientierung
- Orientierung-durchgeführt-Modus.

Wenn ein M19 ausgeführt wird, wechselt die Spindel in den Modus *Suche nach gewünschter Orientierung*, und der HAL-Pin `spindle.__N__.orient` wird aktiviert. Die gewünschte Zielposition wird durch die Pins "spindle.*N*.orient-angle" und "spindle.*N*.orient-fwd" festgelegt und durch die M19-Parameter R und P gesteuert.

Es wird erwartet, dass die HAL-Unterstützungslogik auf `spindle.__N__.orient` reagiert, indem sie die Spindel in die gewünschte Position bewegt. Wenn dies abgeschlossen ist, wird erwartet, dass die HAL-Logik dies durch die Bestätigung des Pins "Spindle.*N*.is-oriented" .

Motion quittiert dies durch Deaktivierung des Pins "Spindle.*N*.orient" und aktiviert den Pin "Spindle.*N*.locked", um den Modus "Orientierung abgeschlossen" anzuzeigen. Außerdem wird der Pin "spindle.*N*.brake" angehoben. Die Spindel befindet sich nun im Modus *Orientierung abgeschlossen*.

Wenn der Pin "spindle.*N*.orient-fault" einen Wert ungleich Null hat, während "spindle.*N*.orient" wahr ist und "spindle.*N*.is oriented" noch nicht aktiviert ist, wird der M19-Befehl abgebrochen, eine Meldung mit dem Fehlercode angezeigt und die Bewegungswarteschlange geleert. Die Spindel kehrt in den Rotationsmodus zurück.

Außerdem kann jeder der Befehle M3, M4 oder M5 entweder den Modus *Suche nach gewünschter Orientierung* oder *Orientierung abgeschlossen* abbrechen. Dies wird durch das Deassertieren der Pins `spindle-orient` und `spindle-locked` angezeigt.

Der Pin "spindle-orient-mode" spiegelt das M19 P-Wort wider und ist wie folgt zu interpretieren:

- 0: Drehen im oder gegen den Uhrzeigersinn für kleinste Winkelbewegung
- 1: immer rot
- 2: immer gegen den Uhrzeigersinn drehen

Sie kann mit der HAL-Komponente "orient" verwendet werden, die einen PID-Befehlswert auf der Grundlage der Spindelgeberposition, des `spindle-orient-angle` ("Spindelorientierungswinkel") and `spindle-orient-mode` ("Spindelorientierungsmodus") liefert.

5.7.3. Achs- und Gelenkpins und Parameter

Diese Pins und Parameter werden durch das Echtzeitmodul *motmod* angelegt. [In *trivial kinematics* Maschinen gibt es eine Eins-zu-Eins-Entsprechung zwischen Gelenken und Achsen.] Sie werden von der Funktion *motion-controller* gelesen und aktualisiert.

Einzelheiten zu den Pins und Parametern finden Sie in der Motion-Manpage *motion(9)*.

5.7.4. iocontrol

iocontrol - nimmt Nicht-Echtzeit-E/A-Befehle via NML entgegen, interagiert mit HAL.

Die HAL-Pins von iocontrol werden im Nicht-Echtzeit-Kontext ein- und ausgeschaltet. Wenn Sie strenge Timing-Anforderungen haben oder einfach mehr E/A benötigen, sollten Sie stattdessen die echtzeitsynchronisierte E/A verwenden, die von motion bereitgestellt wird.

Pins

- *iocontrol.0.coolant-flood* (bool, out) TRUE when flood coolant is requested.
- *iocontrol.0.coolant-mist* (bool, out) TRUE when mist coolant is requested.
- *iocontrol.0.emc-enable-in* (bool, in) Should be driven FALSE when an external E-Stop condition exists.
- *iocontrol.0.tool-change* (bool, out) TRUE when a tool change is requested.
- *iocontrol.0.tool-changed* (bool, in) Should be driven TRUE when a tool change is completed.
- *iocontrol.0.tool-number* (sint, out) The current tool number.
- *iocontrol.0.tool-prep-number* (sint, out) The number of the next tool, from the RS274NGC T-word.
- *iocontrol.0.tool-prepare* (bool, out) TRUE when a tool prepare is requested.
- *iocontrol.0.tool-prepared* (bool, in) Should be driven TRUE when a tool prepare is completed.
- *iocontrol.0.user-enable-out* (bool, out) FALSE when an internal E-Stop condition exists.
- *iocontrol.0.user-request-enable* (bool, out) TRUE when the user has requested that E-Stop be cleared.

5.7.5. INI-Einstellungen

Eine Reihe von INI-Einstellungen werden als HAL Eingangspins zur Verfügung gestellt.

Pins

N bezieht sich auf eine Gelenknummer, *L* auf einen Achsenbuchstaben.

- *ini.N.ferror* - (real, in) [JOINT_N]FERROR
- *ini.N.min_ferror* - (real, in) [JOINT_N]MIN_FERROR
- *ini.N.backlash* - (real, in) [JOINT_N]BACKLASH
- *ini.N.min_limit* - (real, in) [JOINT_N]MIN_LIMIT
- *ini.N.max_limit* - (real, in) [JOINT_N]MAX_LIMIT
- *ini.N.max_velocity* - (real, in) [JOINT_N]MAX_VELOCITY
- *ini.N.max_acceleration* - (real, in) [JOINT_N]MAX_ACCELERATION
- *ini.N.home* - (real, in) [JOINT_N]HOME
- *ini.N.home_offset* - (real, in) [JOINT_N]HOME_OFFSET
- *ini.N.home_offset* - (sint, in) [JOINT_N]HOME_SEQUENCE
- *ini.L.min_limit* - (real, in) [AXIS_L]MIN_LIMIT

- *ini.L.max_limit* - (real, in) [AXIS_L]MAX_LIMIT
- *ini.L.max_velocity* - (real, in) [AXIS_L]MAX_VELOCITY
- *ini.L.max_acceleration* - (real, in) [AXIS_L]MAX_ACCELERATION

NOTE Die achsspezifischen Pins min_limit und max_limit werden nach der Referenzfahrt kontinuierlich berücksichtigt. Die achsspezifischen Pins ferror und min_ferror werden berücksichtigt, wenn die Maschine eingeschaltet ist und sich nicht in Position befindet. Die achsspezifischen Pins max_velocity und max_acceleration werden abgetastet, wenn die Maschine eingeschaltet und der motion_state frei ist (Referenzfahrt oder Jogging), aber nicht, wenn ein Programm läuft (Auto-Modus) oder im MDI-Modus. Folglich hat eine Änderung der Pin-Werte bei laufendem Programm erst dann Auswirkungen, wenn das Programm gestoppt wird und der motion_state wieder frei ist.

- *ini.traj_arc_blend_enable* - (bool, in) [TRAJ]ARC_BLEND_ENABLE
- *ini.traj_arc_blend_fallback_enable* - (bool, in) [TRAJ]ARC_BLEND_FALLBACK_ENABLE
- *ini.traj_arc_blend_gap_cycles* - (real, in) [TRAJ]ARC_BLEND_GAP_CYCLES
- *ini.traj_arc_blend_optimization_depth* - (real, in) [TRAJ]ARC_BLEND_OPTIMIZATION_DEPTH
- *ini.traj_arc_blend_ramp_freq* - (real, in) [TRAJ]ARC_BLEND_RAMP_FREQ

NOTE The traj_arc_blend pins are sampled continuously but changing pin values while a program is running may not have immediate effect due to queueing of commands.

- *ini.traj_default_acceleration* - (real, in) [TRAJ]DEFAULT_ACCELERATION
- *ini.traj_default_velocity* - (real, in) [TRAJ]DEFAULT_VELOCITY
- *ini.traj_max_acceleration* - (real, in) [TRAJ]MAX_ACCELERATION

S-Kurven Trajektorenplanungs-Pins (werden kontinuierlich gesampelt, können sich zur Laufzeit ändern):

- *ini.traj_planner_type* - (sint, in) [TRAJ]PLANNER_TYPE
- *ini.traj_max_jerk* - (real, in) [TRAJ]MAX_LINEAR_JERK

Pro-Achse Jerk-Begrenzungs-Pins (wobei *L* aus {x, y, z, a, b, c, u, v, or w}):

- *ini.L.max_jerk* - (real, in) [AXIS__L_]MAX_JERK

Joint-spezifische Pins zur Begrenzung des Jerks (*N* spezifiziert die Joint-Nummer 0 bis 8):

- *ini.N.max_jerk* - (real, in) [JOINT__N_]MAX_JERK

5.8. HAL-Komponenten Liste

5.8.1. Komponenten

Die meisten Befehle in der folgenden Liste haben ihre eigenen Manpages. Für einige gibt es erweiterte Beschreibungen, für andere nur begrenzte Beschreibungen. Anhand dieser Liste wissen Sie, welche Komponenten existieren, und Sie können `man` *befehlsname* auf Ihrer UNIX-Befehlszeile verwenden, um zusätzliche Informationen zu erhalten. Um die Informationen in der man-Seite anzuzeigen, geben Sie in einem Terminal-Fenster ein:

```
man axis
```

Die eine oder andere Einrichtung eines UNIX-Systems kann die explizite Angabe der Sektion der man-Seite erfordern. Wenn Sie nicht finden, die man-Seite oder der Name der man-Seite ist bereits von einem anderen UNIX-Tool mit der LinuxCNC man-Seite in einem anderen Abschnitt, dann versuchen Sie, explizit den Abschnitt, wie in `man _section-no_ axis`, mit *section-no* = 1 für Nicht-Echtzeit-und 9 für Echtzeit-Komponenten.

NOTE Siehe auch den Abschnitt *Man Pages* auf der Dokumentationsübersicht oder de, Dateibaum. Um in den Man Pages zu suchen, verwenden Sie das UNIX-Tool `apropos`.

Benutzerschnittstellen (Nicht-Echtzeit)

Maschinensteuerung

axis	AXIS LinuxCNC (ehemals "Enhanced Machine Controller") GUI
axis-remote	AXIS Fernzugriff (engl. Remote Interface)
gmoccapy	Touchy LinuxCNC Grafische Benutzeroberfläche
gscreen	Touchy LinuxCNC Grafische Benutzeroberfläche
halui	Beobachten Sie die HAL-Pins und befehlen Sie LinuxCNC über NML
mdro	nur manuell Digital Read Out (DRO)
ngcgui	Framework zur dialogorientierten G-Code-Generierung auf dem Controller
panelui	
pyngcgui	Python-Implementierung von NGCGUI
touchy	AXIS - TOUCHY LinuxCNC Grafische Benutzeroberfläche

Virtuelle Schalttafeln (VCP)

gladevcp	Virtuelles Bedienfeld für LinuxCNC basierend auf Glade, Gtk und HAL Widgets
gladevcp_demo	GladeVCP - verwendet von Beispielkonfigurationen zur Demonstration von Glade Virtual_demo

gremlin_view	Grafische Vorschau des G-Codes
moveoff_gui	GUI für die Moveoff-Komponente
pyui	Dienstprogramm für Panelui
pyvcp	Virtuelles Bedienfeld für LinuxCNC
pyvcp_demo	Python Virtual Control Panel Demonstrationskomponente
qtvcp	Qt-basiertes virtuelles Bedienfeld

Virtuelle Vismach-Maschinen

5axisgui	Vismach Virtuelle Maschine GUI
hbmgui	Vismach Virtuelle Maschine GUI
hexagui	Vismach Virtuelle Maschine GUI
lineardelta	Vismach Virtuelle Maschine GUI
maho600gui	hexagui - Vismach Virtual Machine-GUI
max5gui	hexagui - Vismach Virtual Machine-GUI
melfagui	Vismach Virtuelle Maschine GUI
puma560gui	puma560agui - GUI für virtuelle Maschinen von Vismach
pumagui	Vismach Virtuelle Maschine GUI
rotarydelta	Vismach Virtuelle Maschine GUI
scaragui	Vismach Virtuelle Maschine GUI
xyzac-trt-gui	Vismach Virtuelle Maschine GUI
xyzbc-trt-gui	Vismach Virtuelle Maschine GUI
xyzab-tdr-gui	Vismach Virtuelle Maschine GUI

Bewegung (Nicht-Echtzeit)

io	iocontrol - interagiert mit HAL oder G-Code in Nicht-Echtzeit
iocontrol	Interaktion mit HAL oder G-Code in Nicht-Echtzeit
mdi	Senden von G-Code-Befehlen vom Terminal an die laufende LinuxCNC-Instanz
milltask	Nicht-Echtzeit-Task-Controller für LinuxCNC

Hardware-Treiber

VFD & Kommunikationsschnittstellen (Nicht-Echtzeit)

elbpcom	Kommunikation mit Mesa Ethernet-Karten
gs2_vfd	HAL Nicht-Echtzeit-Komponente für Automation Direct GS2 VFDs
hy_gt_vfd	HAL-Nicht-Echtzeit-Komponente für Huanyang-VFDs der GT-Serie
hy_vfd	HAL Nicht-Echtzeit-Komponente für Huanyang VFDs
mb2hal	MB2HAL ist eine generische Nicht-Echtzeit-HAL-Komponente zur Kommunikation mit einem oder mehreren Modbus-Geräten. Modbus RTU und Modbus TCP werden unterstützt.
mitsub_vfd	HAL Nicht-Echtzeit-Komponente für Mitsubishi A500 F500 E500 A500 D700 E700 F700-Serien VFDs (andere können funktionieren)
monitor-xhc-hb04	Überwacht die XHC-HB04-Hängeleuchte und warnt vor einer Unterbrechung der Verbindung
pi500_vfd	Powtran PI500 Modbus-Treiber
pmx485	Modbus-Kommunikation mit einem Powermax-Plasmaschneidgerät
pmx485-test	Testen der Modbus-Kommunikation mit einem Powermax-Plasmaschneidgerät
shuttle	Steuern Sie HAL-Pins mit den Geräten ShuttleXpress, ShuttlePRO und ShuttlePRO2 von Contour Design
svd-ps_vfd	HAL Nicht-Echtzeit Komponente für SVD-P(S) VFDs
vfdb_vfd	HAL Nicht-Echtzeit-Komponente für Delta VFD-B Frequenzumrichter
vfs11_vfd	HAL Nicht-Echtzeit-Komponente für Toshiba-Schneider VF-S11 Frequenzumrichter
wj200_vfd	Hitachi WJ200 Modbus-Treiber
xhc-hb04	Nicht-Echtzeit-HAL-Komponente für das xhc-hb04-Pendant
xhc-hb04-accels	Veraltetes Skript zum Joggen von Wheel
xhc-whb04b-6	Nicht-Echtzeit drehbares Rad zu Positionierung (engl. jog dial) HAL Komponente für das drahtlose XHC WHB04B-6 USB Gerät

Mesa und andere I/O-Karten (Echtzeit)

hal_ppmc	Pico Systems Treiber für analoge Servo-, PWM- und Stepper-Controller
hal_bb_gpio	Treiber für BeagleBone GPIO-Pins
hal_parport	Echtzeit HAL-Komponente zur Kommunikation mit einem oder mehreren parallelen PC-Ports
hm2_7i43	Mesa Electronics-Treiber für das 7I43 EPP Anything IO Board mit HostMot2 (weitere Informationen finden Sie in der Manpage)

hm2_7i90	LinuxCNC HAL-Treiber für die Mesa Electronics 7I90 EPP Anything IO-Karte mit HostMot2-Firmware
hm2_eth	LinuxCNC HAL-Treiber für die Mesa Electronics Ethernet Anything IO-Karten, mit HostMot2-Firmware
hm2_pci	Mesa Electronics-Treiber für die 5I20-, 5I22-, 5I23-, 4I65- und 4I68 Anything I/O-Karten mit HostMot2-Firmware. (Siehe die Manpage für weitere Informationen)
hm2_rpspi	LinuxCNC HAL-Treiber für die Mesa Electronics SPI Anything IO Boards, mit HostMot2-Firmware
hm2_spi	LinuxCNC HAL-Treiber für die Mesa Electronics SPI Anything IO Boards, mit HostMot2-Firmware
hostmot2	Mesa Electronics Treiber für die HostMot2-Firmware.
max31855	Unterstützung für den MAX31855 Thermoelement-zu-Digital-Wandler mit Bitbanged SPI
mesa_7i65	Mesa Electronics-Treiber für die 7I65 Acht-Achsen-Servokarte. (Siehe die Manpage für weitere Informationen)
mesa_pktgyro_test	Einfacher PktUART-Test mit Microstrain 3DM-GX3-15 Kreisel
mesa_uart	Eine Beispielkomponente, die zeigt, wie man auf den Hostmot2 UART zugreift
serport	Hardwaretreiber für die digitalen E/A-Bits der seriellen Schnittstelle des 8250 und 16550
setsserial	Ein Dienstprogramm zur Einstellung der Smart Serial NVRAM-Parameter
sserial	hostmot2 - Smart Serial LinuxCNC HAL Treiber für die Mesa Electronics HostMot2 Smart-Serial Remote Karten

Dienstprogramme (Nicht-Echtzeit)

hal-histogram	Plottet den Wert eines HAL-Pins als Histogramm
halcompile	Erstellen, kompilieren und installieren von LinuxCNC HAL Komponenten
halmeter	Beobachten von HAL-Pins, -Signale und -Parametern
halcmd	Manipuliert des LinuxCNC HAL von der Kommandozeile
halcmd_twopass	Hilfsskript zum Parsen von HAL-Dateien. Es ermöglicht die Verwendung mehrerer **load**-Befehle für mehrere Instanzen derselben Komponente.
halreport	Erzeugt einen Bericht über den Status des HAL
halrmt	Fernsteuerungs-Schnittstelle für LinuxCNC
halrun	Manipuliert des LinuxCNC HAL von der Kommandozeile

halsampler	Probendaten von HAL in Echtzeit
halscope	Software-Oszilloskop zur Anzeige von Echtzeit-Wellenformen von HAL-Pins und -Signalen
halshow	HAL-Parameter, Pins und Signale anzeigen
halstreamer	Streamen von Daten aus Dateien an HAL in Echtzeit
haltcl	Manipuliert die LinuxCNC HAL von der Kommandozeile aus mit Tcl
image-to-gcode	Konvertiert Bitmap-Bilder in G-Code
inivar	Abfragen einer INI-Dateie
latency-histogram	Plottet Histogramm der Maschinenlatenz
latency-plot	Eine weitere Möglichkeit, Latenzzahlen anzuzeigen
latency-test	Testen der Latenzzeit des Echtzeitsystems
linuxcncmkdesktop	Anlegen eines Desktop-Symbols für LinuxCNC
motion-logger	Log Bewegungsbefehle (engl. log motion commmands) gesendet von LinuxCNC
pncconf	Konfigurationsassistent für Mesa-Karten
sim_pin	GUI zur Anzeige und Einstellung von einem oder mehreren HAL-Eingängen
stepconf	Ein Konfigurationsassistent für Maschinen mit parallelen Anschlüssen
update_ini	Konvertiert INI-Dateien im 2.7-Format in das 2.8-Format
debuglevel	Setzt den Debuglevel für den Nicht-Echtzeit-Teil von LinuxCNC
emccalib	INI-Tuning-Variablen während des Betriebs anpassen, mit Speicheroption
hal_input	Steuern von HAL-Pins mit jedem Linux-Eingabegerät, einschließlich USB-HID-Geräten
linuxcnc_info	Sammelt Informationen über die LinuxCNC-Version und den host
linuxcnc_module_helper	Steuert den Root-Zugriff auf die Systemhardware
linuxcnc_var	Ruft die LinuxCNC-Variablen ab
linuxcnc	LinuxCNC (ehemals "Enhanced Machine Controller")
linuxcnclcd	LinuxCNC Grafische Benutzeroberfläche für LCD-Zeichenanzeige
linuxcncrsh	Text-Mode-Schnittstelle für die Steuerung von LinuxCNC über das Netzwerk
linuxcncsvr	Ermöglicht Netzwerkzugriff auf LinuxCNC Interna über NML

linuxcnctop	Live LinuxCNC Statusbeschreibung
rs274	Eigenständiger (engl. standalone) G-Code Interpreter
schedrmt	Telnet-basierter Scheduler für LinuxCNC
setup_designer	Ein Skript um das System für Qt Designer zu konfigurieren
teach-in	Manuell die Maschine in eine bestimmte Position bringen und den Status aufzeichnene
tool_mmap_read	Eine Komponente des Werkzeugdatenbanksystems (eine Alternative zur klassischen Werkzeugtabelle)
tool_watch	Eine Komponente des Werkzeugdatenbanksystems (eine Alternative zur klassischen Werkzeugtabelle)
tooledit	Werkzeugtabellen-Editor

Signalverarbeitung (Echtzeit)

Logik und Bitwise

and2	UND-Gatter mit zwei Eingängen. Damit der Ausgang wahr ist, müssen beide Eingänge wahr sein. (and2)
bitwise	Berechnet verschiedene bitweise Operationen an den beiden Eingabewerten
dbounce	Filtert verrauschte digitale Eingänge: Details
debounce	Filtert verrauschte digitale Eingänge: Details Beschreibung
demux	Auswahl eines von mehreren Ausgangsstiften durch Ganzzahl und/oder oder einzelne Bits
demux_generic	Routes a single input signal to one of multiple outputs.
edge	Kanten-Detektor
estop_latch	Notaus-Verriegelung
flipflop	D-Typ Flip-Flop
logic	Allgemeine logische Funktionskomponente
lut5	Logikfunktion mit 5 Eingängen, die auf einer Look-up-Tabelle basiert, siehe Beschreibung
match8	8-Bit-Binär-Match-Detektor
multiclick	Einzel-, Doppel-, Dreifach- und Vierfach-Klick-Detektor
multiswitch	Schaltet zwischen einer bestimmten Anzahl von Ausgangsbits um
not	Inverter

oneshot	One-Shot-Pulsgenerator
or2	ODER-Gatter mit zwei Eingängen
output_buffer	Feed through multiple bits when enable pin is set
reset	Setzt ein IO-Signal zurück
select8	8-Bit-Binär-Match-Detektor.
tap	Tap into a HAL boolean signal to override it from a UI
tof	IEC TOF Timer - Verzögerung der fallenden Flanke eines Signals
toggle	Push-on, push-off von Drucktastern mit kurzem Tastendruck
toggle2nist	Button auf Nist-Logik umschalten
ton	IEC TON Timer - Verzögerung der steigenden Flanke eines Signals
timedelay	Äquivalent eines zeitverzögerten Relais.
tp	IEC TP Timer - erzeugt einen High-Impuls von definierter Dauer bei steigender Flanke
tristate_bit	Legt ein Signal nur dann auf einen E/A-Pin, wenn es aktiviert ist, ähnlich wie ein Tristate-Puffer in der Elektronik
tristate_float	Legt ein Signal nur dann auf einen E/A-Pin, wenn es aktiviert ist, ähnlich wie ein Tristate-Puffer in der Elektronik
xor2	XOR-Gatter mit zwei Eingängen (Exklusiv-ODER)

Arithmetisch und Fließkomma

abs_sint	Berechnet den Absolutwert und das Vorzeichen des Ganzzahl-Eingangssignals
abs	Computes the absolute value and sign of a real input signal
biquad	Biquad IIR-Filter
blend	Lineare Interpolation zwischen zwei Werten durchführen
comp	Komparator mit zwei Eingängen und Hysterese
counter	Zählt Eingangsimpulse (veraltet). Verwenden Sie die Komponente encoder.
ddt	Berechnet die Ableitung der Eingangsfunktion.
deadzone	Gibt den Mittelpunkt zurück, wenn er sich innerhalb des Schwellenwerts befindet.
div2	Quotient aus zwei Fließkomma-Eingaben.
hypot	Rechner für die Hypotenuse (euklidischer Abstand) mit drei Eingaben.
ilowpass	Tiefpassfilter mit ganzzahligen Ein- und Ausgängen
integ	Integrator

invert	Berechnet die Umkehrung des Eingangssignals.
filter_kalman	Eindimensionaler Kalman-Filter, auch bekannt als lineare quadratische Schätzung (LQE)
knob2float	Converts counts (probably from an encoder) to a real value.
led_dim	HAL Komponente zum Dimmen on LEDs
lowpass	Tiefpassfilter
limit1	Begrenzt das Ausgangssignal auf einen Wert zwischen min und max. [2]
limit2	Begrenzt das Ausgangssignal auf den Bereich zwischen min und max. Begrenzt seine Anstiegsgeschwindigkeit auf weniger als maxv pro Sekunde. [3]
limit3	Begrenzen Sie das Ausgangssignal so, dass es zwischen min und max liegt. Begrenzen Sie die Anstiegsrate auf weniger als maxv pro Sekunde. Begrenzung der zweiten Ableitung auf weniger als MaxA pro Sekunde zum Quadrat [4].
lincurve	Eindimensionale Nachschlagetabelle (engl. lookup table)
maj3	Berechne die Mehrheit von 3 Eingaben
minmax	Verfolgt die minimalen und maximalen Werte der Eingabe für die Ausgänge.
mult2	Produkt aus zwei Eingaben.
mux16	Wahl zwischen zwei Eingangswerten aus (multiplexer).
mux2	Wahl zwischen zwei Eingangswerten aus (multiplexer).
mux4	Wählt einen von vier Eingangswerten (multiplexer).
mux8	Wahl aus einem von acht Eingangswerten (multiplexer).
mux_generic	Select one from several inputs and forwards it to a single output (multiplexer).
near	Bestimmt, ob zwei Werte annähernd gleich sind.
offset	Fügt einer Eingabe einen Offset hinzu und subtrahiert ihn vom Feedbackwert.
safety_latch	Riegel (engl. latch) für Fehler-Signale
sample_hold	Probenahme und Halten.
scaled_sint_sums	Summe von vier Eingängen (jeweils mit einer Skala)
scale	Wendet eine Skalierung und einen Offset auf seinen Eingang an.
sum2	Summe aus zwei Eingängen (jeweils mit einem Verstärkungsfaktor) und einem Offset.
time	Kumulierte Laufzeit des Timers zählt HH:MM:SS des *aktiven* Eingangs.
timedelta	Komponente, die das Zeitverhalten bei der Thread-Planung misst.
updown	Zählt aufwärts oder abwärts, mit optionalen Grenzen und Wraparound-Verhalten.
wcomp	Fenster-Komparator.

watchdog	Überwachen Sie einen bis zweiunddreißig Eingänge auf einen „Herzschlag“.
weighted_sum	Gewichtete Summe, konvertiert eine Gruppe von Bits in eine ganze Zahl.
xhc_hb04_util	xhc-hb04 Komfort-Dienstprogramm

Signal-Erzeugung (Echtzeit)

charge_pump	Erzeugt eine Rechteckwelle für den *charge pump*-Eingang einiger Controller-Karten.
pwmgen	Software-PWM/PDM-Erzeugung, siehe Beschreibung.
siggen	Signalgenerator, siehe Beschreibung.
sim_encoder	Simulierter Quadratur-Encoder, siehe Beschreibung.
stepgen	Software-Schrittimpulsgenerierung, siehe Beschreibung.

Typumwandlung

bin2gray	Konvertierung einer Zahl in die Gray-Code Repräsentation
bitmerge	Konvertiert einzelne Eingabe-Bits in einen 32 Bit Wert ohne Vorzeichen
bitslice	Konvertiert eine vorzeichenlose 32-Eingabe in einzelne Bits
conv_bool_real	Converts from bool to real
conv_bool_sint	Convert a value from bool to sint
conv_bool_uint	Convert a value from bool to uint
conv_real_sint	Convert a value from real to sint
conv_real_uint	Convert a value from real to uint
conv_sint_bool	Convert a value from sint to bool
conv_sint_real	Convert a value from sint to real
conv_sint_uint	Convert a value from sint to uint

conv_uint_bool	Convert a value from uint to bool
conv_uint_real	Convert a value from uint to real
conv_uint_sint	Convert a value from uint to sint
gray2bin	Konvertiert Gray-Code-Eingabe in Binärformat

Kinematiken (Echtzeit)

corexy_by_hal	CoreXY-Kinematiken
differential	Kinematik für ein Differentialgetriebe
gantry	LinuxCNC HAL-Komponente für den Antrieb mehrerer Gelenke von einer einzigen Achse
gantrykins	Kinematikmodul, das eine Achse auf mehrere Gelenke abbildet.
genhexkins	Ergibt sechs Freiheitsgrade in Position und Orientierung (XYZABC). Die Position der Motoren wird zur Kompilierzeit festgelegt.
genserkins	Kinematik, die einen allgemeinen Manipulator mit seriellen Gliedern und bis zu 6 Winkelgelenken modellieren kann.
gentrivkins	1:1-Entsprechung zwischen Gelenken und Achsen. Die meisten Standardfräs- und -drehmaschinen verwenden das triviale Kinematikmodul.
kins	Kinematik Definitionen für LinuxCNC.
lineardeltakins	Kinematik für pumaähnliche Roboter
matrixkins	Kalibrierte Kinematik für 3-Achsen-Maschines
maxkins	Kinematik für eine 5-Achsen-Tischfräse namens *max* mit schwenkbarem Kopf (B-Achse) und horizontaler Drehung auf dem Tisch (C-Achse). Ermöglicht UVW-Bewegung im gedrehten Koordinatensystem.
millturn	Umschaltbare Kinematik für eine Fräs-Dreh-Maschine
pentakins	
pumakins	Kinematik für PUMA-ähnliche Roboter.
rosekins	Kinematik für einen Rosenmotor
rotatekins	Die X- und Y-Achsen sind um 45 Grad gegenüber den Gelenken 0 und 1 gedreht.
scarakins	Kinematik für SCARA-Roboter.

switchkinscomp	Switchable kinematics module template
three21kins	Analytical kinematics solver for 6-DOF arm + wrist robots.
tripodkins	Die Gelenke stellen den Abstand des kontrollierten Punktes von drei vordefinierten Orten (den Motoren) dar, was drei Freiheitsgrade in der Position (XYZ) ergibt.
userkins	Vorlage für benutzerdefinierte Kinematiken
xyzab_tdr_kins	Umschaltbare Kinematik für 5-Achsen Maschine mit Drehtischen A und B
xyzacb_trsrn	Umschaltbare Kinematik für eine 6-Achsen Maschine mit Rundtisch C, Spindelrotation (engl. rotary spindle) B und Taumelspindel (engl. nutating spindle) A
xyzbca_trsrn	Umschaltbare Kinematik für eine 6-Achsen Maschine mit Rundtisch C, Spindelrotation (engl. rotary spindle) B und Taumelspindel (engl. nutating spindle) A

Motorsteuerung (engl. motor control) (Echtzeit)

feedcomp	Multipliziert die Eingabe mit dem Verhältnis von aktueller Geschwindigkeit und Vorschubgeschwindigkeit.
homecomp	Referenzfahrt-Modulvorlage
limit_axis	Achseinschränkungen mit dynamischem Bereich
motion	Akzeptiert NML-Bewegungsbefehle, interagiert mit HAL in Echtzeit
simple_tp	Diese Komponente ist ein einachsiger einfacher Trajektorienplaner, wie er auch für das Jogging in LinuxCNC verwendet wird.
tpcomp	Trajectory Planning (tp)-Modulskelett

Motor control (Echtzeit)

at_pid	Proportional-/Integral-/Ableitungsregler mit Selbstoptimierung.
bldc	BLDC- und AC-Servo-Regelkomponenten
clarke2	Zwei-Eingabe-Version der Clarke-Transformation
clarke3	Clarke-Transformation (von 3-Phasen nach kartesisch)
clarkeinv	Inverse Clarke-Transformation
encoder	Software-Zählung von Quadratur-Encoder-Signalen, siehe Description.
pid	Proportionaler/integraler/derivativer Regler, Beschreibung.
pwmgen	Software-PWM/PDM-Erzeugung, siehe Beschreibung.
stepgen	Software-Schrittimpulsgenerierung, siehe Beschreibung.

Simulation/Testen

axistest	Wird verwendet, um eine Achse zu testen. Verwendet in PnCConf.
rtapi_app	schafft eine simulierte Echtzeitumgebung
sim-torch	Ein simulierter Plasmabrennerh
sim_axis_hardware	Eine Komponente zur Simulation von Referenz- und Endschalterns
sim_home_switch	Referenzschalters Simulator
sim_matrix_kb	Konvertierung der HAL-Pin-Eingänge in Tastencodess
sim_parport	Eine Komponente zur Simulation der Pins der hal_parport Komponente
sim_spindle	Simulierte Spindel mit Indeximpulse
simulate_probe	Simulieren eines Sondeneingangs

Sonstiges (Echtzeit)

anglejog	Zwei Achsen (oder Gelenke) in einem Winkel bewegen
classicladder	Echtzeit-Software-SPS (engl. PLC), die auf Kontaktplan-Logik basiert. Siehe Kapitel ClassicLadder für weitere Informationen.
charge_pump	Erzeugt eine Rechteckwelle für den *charge pump*-Eingang einiger Controller-Karten.
encoder_ratio	Elektronisches Getriebe zur Synchronisierung zweier Achsen.
enum	Zerlegen einer Ganzzahl in einzelne Bits
eoffset_per_angle	Externen Versatz pro Winkel berechnen
GladeVCP (Echtzeit)	zeigt mit GTK/Glade erstellte virtuelle Kontrollfelder an
histobins	Dienstprogramm für Histogramm-Bins für `scripts/hal-histogram`
joyhandle	Setzt nichtlineare Joypad-Bewegungen, Deadbands und Skalen.
latencybins	Berechnungshilfen für scrips/latency-histogram
message	Anzeigen einer Meldung
moveoff	Komponente für HAL-only Offsets
raster	Ausgabe der Laserleistung basierend auf vorprogrammierten Rasterdaten

sampler	Probendaten von HAL in Echtzeit.
siggen	Signalgenerator, siehe Beschreibung.
sphereprobe	Sondieren einer angenommenen Hemisphäre.
threads	Erzeugt harte Echtzeit-HAL-Threads.
threadtest	Komponente zum Testen des Threadverhaltens.
steptest	Wird von StepConf verwendet, um das Testen von Beschleunigungs- und Geschwindigkeitswerten für eine Achse zu ermöglichen.
streamer	Streamen von Daten aus Dateien an HAL in Echtzeit.
supply	Legen Sie Ausgabepins mit Werten aus Parametern fest (veraltet).

Andere Hardware-Schnittstellen (Echtzeit)

laserpower	Skaliert die Laserausgangsleistung auf der Grundlage der Eingangsleistung der Geschwindigkeit und der zurückzulegenden Entfernung
lcd	Streamen von HAL-Daten auf einen LCD-Bildschirm
matrix_kb	Konvertierung von Ganzzahlen in HAL-Pins. Optionales Scannen einer Matrix von IO-Ports um diese Ganzzahlen zu erzeugen.

Mit Bezug zur Spindel

gearchange	Wählt einen von zwei Geschwindigkeitsbereichen aus.
orient	Bereitstellung eines PID-Befehlseingangs für den Orientierungsmodus auf der Grundlage der aktuellen Spindelposition, des Zielwinkels und des Orientierungsmoduse
spindle	Steuerung einer Spindel mit unterschiedlicher Beschleunigung und Verzögerung und optionaler Schaltskalierung
spindle_monitor	Erkennung von Erreichen der Spindeldrehzahl und Unterdrehzahlen

Werkzeug-bezogen

carousel	Orientiert ein Werkzeugwechsler-Karussell anhand verschiedener Kodierungsschematas
hal_manualtoolchange	HAL nicht-Echtzeit-Komponente, um manuelle Werkzeugwechsel zu ermöglichen.

Bezogen auf Plasmaschneider

thc	Brennerhöhensteuerung mit einer Mesa THC-Karte oder einem beliebigen Analog-/Geschwindigkeitseingang
thcud	Brennerhöhensteuerung Auf/Ab-Eingang
ohmic	LinuxCNC HAL-Komponente, die eine Mesa THCAD (A/D Karte) für Ohmic Sensing verwendet
plasmac	Eine Steuerung für Plasmaschneider

5.8.2. Not categorized (auto generated from man pages)

axis	
histobinstream	histogram bins utility for scripts/hal-histogram
hm2_modbus	A hostmot2 driver that implements the Modbus protocol using the PktUART ports.
hm2_spix	LinuxCNC HAL driver for the Mesa Electronics Anything IO boards with SPI enabled HostMot2 firmware.
inivalue	Abfragen einer INI-Dateie
joint_axis_mapper	Translate faults from Joint to Axis
latencybinstream	comp utility for scripts/latency-histogram.py
linuxcnc_check_ini	LinuxCNC INI-file configuration checker
mesambccc	Utility for compiling hm2_modbus command control description files
millturn	millturn, millturngui - Vismach Virtuelle Maschine GUI
millturngui	
momentary2nist	momentary button to nist logic
mqtt-publisher	send HAL pin data to MQTT broker periodically
mtconnect-agent	expose LinuxCNC machine status and kinematics over MTConnect
pushmsg	Push configurable RT messages to non-RT logging on pin change
qtplasmac-materials	Create a plasma materials file.

qtplasmac_gcode	Python script shipping with Plasmac, a Plasma cutting system.
radiobutton	Select one from several outputs with momentary switches
scorbot-er-3	to link the Intellitek Scorbot educational robot to LinuxCNC
sendkeys	send input events based on pins or scancodes from HAL
thermistor	compute temperature indicated by a thermistor
three21kins	
trivkins	

5.8.3. Without man page or broken link (auto generated from component list)

hal_ppmc

5.8.4. HAL-API-Aufrufe

- hal(3)
- hal_add_funct_to_thread(3)
- hal_bool_t(3)
- hal_comp_by_id(3)
- hal_comp_by_name(3)
- hal_create_thread(3)
- hal_del_funct_from_thread(3)
- hal_del_init_funct_from_thread(3)
- hal_exit(3)
- hal_export_funct(3)
- hal_export_functf(3)
- hal_get_bool(3)
- hal_get_lock(3)
- hal_get_p(3)
- hal_get_real(3)
- hal_get_realtime_type(3)
- hal_getref_p(3)
- hal_getref_s(3)
- hal_get_s(3)
- hal_get_si32(3)

- hal_get_si32_clamped(3)
- hal_get_sint(3)
- hal_getter(3)
- hal_get_ui32(3)
- hal_get_ui32_clamped(3)
- hal_get_uint(3)
- hal_init(3)
- hal_init_funct_to_thread(3)
- hal_link(3)
- hal_list_comp(3)
- hal_list_funct(3)
- hal_list_p(3)
- hal_list_p_s(3)
- hal_list_s(3)
- hal_list_thread(3)
- hal_malloc(3)
- hal_param_alias(3)
- hal_param_new(3)
- hal_param_new_bool(3)
- hal_param_new_fake(3)
- hal_param_new_real(3)
- hal_param_new_sint(3)
- hal_param_new_uint(3)
- hal_parport(3)
- hal_pdir_t(3)
- hal_pin_alias(3)
- hal_pin_new(3)
- hal_pin_new_bool(3)
- hal_pin_new_port(3)
- hal_pin_new_real(3)
- hal_pin_new_sint(3)
- hal_pin_new_uint(3)
- hal_port(3)
- hal_port_buffer_size(3)

- hal_port_clear(3)
- hal_port_peek(3)
- hal_port_peek_commit(3)
- hal_port_read(3)
- hal_port_readable(3)
- hal_port_t(3)
- hal_port_writable(3)
- hal_port_write(3)
- hal_query_t(3)
- hal_ready(3)
- hal_real_t(3)
- hal_refs_u(3)
- hal_set_bool(3)
- hal_set_constructor(3)
- hal_set_lock(3)
- hal_set_p(3)
- hal_set_real(3)
- hal_set_s(3)
- hal_set_si32(3)
- hal_set_sint(3)
- hal_setter(3)
- hal_set_ui32(3)
- hal_set_uint(3)
- hal_signal_delete(3)
- hal_signal_new(3)
- hal_sint_t(3)
- hal_start_threads(3)
- hal_statistics(3)
- hal_stream(3)
- hal_type_t(3)
- hal_uint_t(3)
- hal_unlink(3)
- hal_unready(3)

5.8.5. RTAPI-Aufrufe

Helpers:

- EXPORT_FUNCTION(3)
- MODULE_AUTHOR(3)
- MODULE_DESCRIPTION(3)
- MODULE_LICENSE(3)
- RTAPI_MP_ARRAY_INT(3)
- RTAPI_MP_ARRAY_LONG(3)
- RTAPI_MP_ARRAY_STRING(3)
- RTAPI_MP_INT(3)
- RTAPI_MP_LONG(3)
- RTAPI_MP_STRING(3)

API calls:

- rtapi(3)
- rtapi_app_exit(3)
- rtapi_app_main(3)
- rtapi_atomic(3)
- rtapi_bool(3)
- rtapi_byteorder(3)
- rtapi_clock_set_period(3)
- rtapi_delay(3)
- rtapi_delay_max(3)
- rtapi_device(3)
- rtapi_div_s64(3)
- rtapi_div_s64_rem(3)
- rtapi_div_u64(3)
- rtapi_div_u64_rem(3)
- rtapi_exit(3)
- rtapi_firmware(3)
- rtapi_get_msg_level(3)
- rtapi_get_time(3)
- rtapi_gfp(3)
- rtapi_inb(3)

- rtapi_init(3)
- rtapi_io(3)
- rtapi_is(3)
- rtapi_list(3)
- rtapi_module_param(3)
- rtapi_mutex(3)
- rtapi_open_as_root(3)
- rtapi_outb(3)
- rtapi_parport(3)
- rtapi_pci(3)
- rtapi_print(3)
- rtapi_print_msg(3)
- rtapi_prio(3)
- rtapi_prio_highest(3)
- rtapi_prio_lowest(3)
- rtapi_prio_next_higher(3)
- rtapi_prio_next_lower(3)
- rtapi_region(3)
- rtapi_release_region(3)
- rtapi_request_region(3)
- rtapi_set_msg_level(3)
- rtapi_shmem(3)
- rtapi_shmem_delete(3)
- rtapi_shmem_getptr(3)
- rtapi_shmem_new(3)
- rtapi_slab(3)
- rtapi_snprintf(3)
- rtapi_stdint(3)
- rtapi_string(3)
- rtapi_strlcpy(3)
- rtapi_task_delete(3)
- rtapi_task_new(3)
- rtapi_task_pause(3)
- rtapi_task_resume(3)

- rtapi_task_self(3)
- rtapi_task_start(3)
- rtapi_task_wait(3)
- rtapi_vsnprintf(3)

5.9. Beschreibungen der HAL-Komponenten

Dieses Kapitel enthält Details zu den Kernfunktionen von LinuxCNC, die ein genaues Timing für

- die Erzeugung von Signalen, die von der Hardware (z. B. Motoren) interpretiert werden, oder
- für die Interpretation der von der Hardware gesendeten Signale (z. B. Encoder).

5.9.1. StepGen

Diese Komponente ermöglicht die softwarebasierte Erzeugung von Schrittimpulsen als Reaktion auf Positions- oder Geschwindigkeitsbefehle. Im Positionsmodus verfügt sie über eine integrierte, voreingestellte Positionsschleife, so dass eine PID-Einstellung nicht erforderlich ist. Im Geschwindigkeitsmodus treibt sie einen Motor mit der befohlenen Geschwindigkeit an, wobei Geschwindigkeits- und Beschleunigungsgrenzen eingehalten werden. Es handelt sich um eine reine Echtzeitkomponente, die je nach CPU-Geschwindigkeit usw. maximale Schrittfrequenzen von 10 kHz bis vielleicht 50 kHz erreichen kann. Das Blockdiagramm des Schrittimpulsgenerators zeigt drei Blockdiagramme, jedes davon ist ein einzelner Schrittimpulsgenerator. Das erste Diagramm ist für den Schritttyp *0* (Schritt und Richtung). Das zweite ist für den Schritttyp *1* (Auf/Ab oder Pseudo-PWM), und das dritte für die Schritttypen 2 bis 14 (verschiedene Schrittmuster). Die ersten beiden Diagramme zeigen die Steuerung im Positionsmodus, das dritte den Geschwindigkeitsmodus. Steuermodus und Schrittart werden unabhängig voneinander eingestellt, und es kann jede beliebige Kombination gewählt werden.

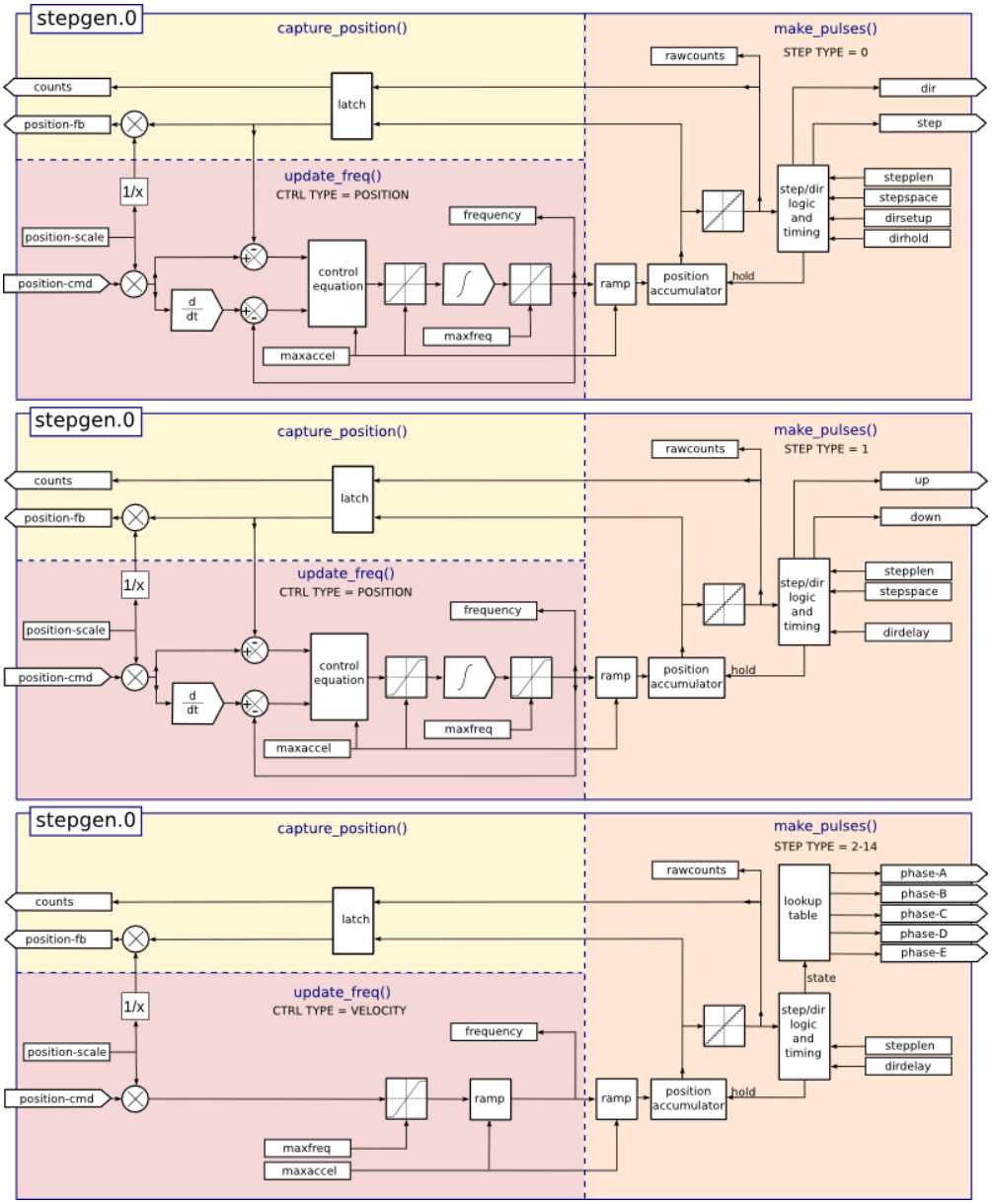


Figure 93. Schrittimpulsgenerator-Blockdiagramm Positionsmodus

Laden der Komponente `stepgen`

```
halcmd: loadrt stepgen step_type=<type-array> [ctrl_type=<ctrl_array>]
```

<type-array>

ist eine Reihe von durch Kommata getrennten Dezimalzahlen. Jede Zahl bewirkt, dass ein Einzelschritt-Impulsgenerator geladen wird; der Wert der Zahl bestimmt die Schrittart.

<ctrl_array>

ist eine durch Komma getrennte Folge von *p*- oder *v*-Zeichen, um den Positions- oder Geschwindigkeitsmodus anzugeben.

ctrl_type

ist optional, wenn sie weggelassen wird, werden alle Schrittgeneratoren im Positionsmodus arbeiten.

Zum Beispiel:

```
halcmd: loadrt stepgen step_type=0,0,2 ctrl_type=p,p,v
```

Will install three step generators. The first two use step type *0* (step and direction) and run in position mode. The last one uses step type *2* (quadrature) and runs in velocity mode. There is no default value for <*config-array*>; at least one step type must be given or loading fails. The maximum number of step generators is 16 (as defined by MAX_CHAN in stepgen.c). Each generator is independent, but all are updated by the same function(s) at the same time. In the following descriptions, <*chan*> is the number of a specific generator. The first generator is number 0.

Komponente "stepgen" entfernen (engl. unload)

```
halcmd: unloadrt stepgen
```

Pins

Zur gewählten Schrift- und Steuerungsart.

- (real) stepgen.`__<chan>__.position-cmd` - Desired motor position, in position units (position mode only).
- (real) stepgen.`__<chan>__.velocity-cmd` - Desired motor velocity, in position units per second (velocity mode only).
- (sint) stepgen.`__<chan>__.counts` - Feedback position in counts, updated by *capture_position()*.
- (real) stepgen.`__<chan>__.position-fb` - Feedback position in position units, updated by *capture_position()*.
- (bool) stepgen.`__<chan>__.enable` - Enables output steps - when false, no steps are generated.
- (bool) stepgen.`__<chan>__.step` - Step pulse output (step type 0 only).
- (bool) stepgen.`__<chan>__.dir` - Direction output (step type 0 only).
- (bool) stepgen.`__<chan>__.up` - UP pseudo-PWM output (step type 1 only).
- (bool) stepgen.`__<chan>__.down` - DOWN pseudo-PWM output (step type 1 only).
- (bool) stepgen.`__<chan>__.phase-A` - Phase A output (step types 2-15 only).
- (bool) stepgen.`__<chan>__.phase-B` - Phase B output (step types 2-15 only).
- (bool) stepgen.`__<chan>__.phase-C` - Phase C output (step types 3-15 only).
- (bool) stepgen.`__<chan>__.phase-D` - Phase D output (step types 5-15 only).
- (bool) stepgen.`__<chan>__.phase-E` - Phase E output (step types 11-15 only).

Parameter

- (real) stepgen.`__<chan>__.position-scale` - Steps per position unit. This parameter is used for both output and feedback.
- (real) stepgen.`__<chan>__.maxvel` - Maximum velocity, in position units per second. If 0.0, has

no effect.

- (real) `stepgen.`__<chan>__.maxaccel` - Maximum accel/decel rate, in positions units per second squared. If 0.0, has no effect.
- (real) `stepgen.`__<chan>__.frequency` - The current step rate, in steps per second.
- (uint) `stepgen.`__<chan>__.steplen` - Length of a step pulse (step type 0 and 1) or minimum time in a given state (step types 2-15), in nano-seconds.
- (uint) `stepgen.`__<chan>__.stepspace` - Minimum spacing between two step pulses (step types 0 and 1 only), in nano-seconds. Set to 0 to enable the stepgen *doublefreq* function. To use *doublefreq* the parport reset function must be enabled.
- (uint) `stepgen.`__<chan>__.dirsetup` - Minimum time from a direction change to the beginning of the next step pulse (step type 0 only), in nanoseconds.
- (uint) `stepgen.`__<chan>__.dirhold` - Minimum time from the end of a step pulse to a direction change (step type 0 only), in nanoseconds.
- (uint) `stepgen.`__<chan>__.dirdelay` - Minimum time any step to a step in the opposite direction (step types 1-15 only), in nano-seconds.
- (sint) `stepgen.`__<chan>__.rawcounts` - The raw feedback count, updated by *make_pulses()*.

Im Positionsmodus werden die Werte von maxvel und maxaccel von der internen Positionsschleife verwendet, um Schrittimpulsfolgen zu vermeiden, denen der Motor nicht folgen kann. Wenn sie auf Werte eingestellt sind, die für den Motor geeignet sind, führt selbst eine große momentane Änderung der befohlenen Position zu einer sanften trapezförmigen Bewegung zur neuen Position. Der Algorithmus misst sowohl den Positions- als auch den Geschwindigkeitsfehler und berechnet eine Beschleunigung, die versucht, beide gleichzeitig auf Null zu reduzieren. Weitere Einzelheiten, einschließlich des Inhalts des Feldes "Kontrollgleichung" (engl. control equation), finden Sie im Code.

Im Geschwindigkeitsmodus ist maxvel ein einfacher Grenzwert, der auf die befohlene Geschwindigkeit angewendet wird, und maxaccel wird verwendet, um die tatsächliche Frequenz zu rampen, wenn sich die befohlene Geschwindigkeit abrupt ändert. Wie im Positionsmodus sorgen die richtigen Werte für diese Parameter dafür, dass der Motor der erzeugten Impulsfolge folgen kann.

Schrittypen

Step generator supports 16 different *step sequences* (types 0 to 15):

Schritttyp 0 (engl. step type 0)

Schritttyp 0 ist der Standard-Schritt- und Richtungstyp. Bei der Konfiguration für den Schritttyp 0 gibt es vier zusätzliche Parameter, die das genaue Timing der Schritt- und Richtungssignale bestimmen. In der folgenden Abbildung ist die Bedeutung dieser Parameter dargestellt. Die Parameter sind in Nanosekunden angegeben, werden aber auf ein ganzzahliges Vielfaches der Thread-Periode für den Thread aufgerundet, der *make_pulses()* aufruft. Wenn zum Beispiel *make_pulses()* alle 16 µs aufgerufen wird und steplen 20000 ist, dann sind die Schrittimpulse 2 x 16 = 32 µs lang. Der Standardwert für alle vier Parameter ist 1 ns, aber die automatische Rundung tritt in Kraft, wenn der Code zum ersten Mal ausgeführt wird. Da ein Schritt *steplen* ns hoch und *stepspace* ns niedrig benötigt, ist die maximale Frequenz 1.000.000.000 geteilt durch (*steplen* + *stepspace*)'. Wenn *maxfreq* höher als dieser Grenzwert

eingestellt ist, wird er automatisch gesenkt. Ist *maxfreq* gleich Null, bleibt er Null, aber die Ausgangsfrequenz wird trotzdem begrenzt.

Bei Verwendung des Parallelport-Treibers kann die Schrittfrequenz mit der Funktion parport reset in Verbindung mit der Einstellung *doublefreq* von StepGen verdoppelt werden.

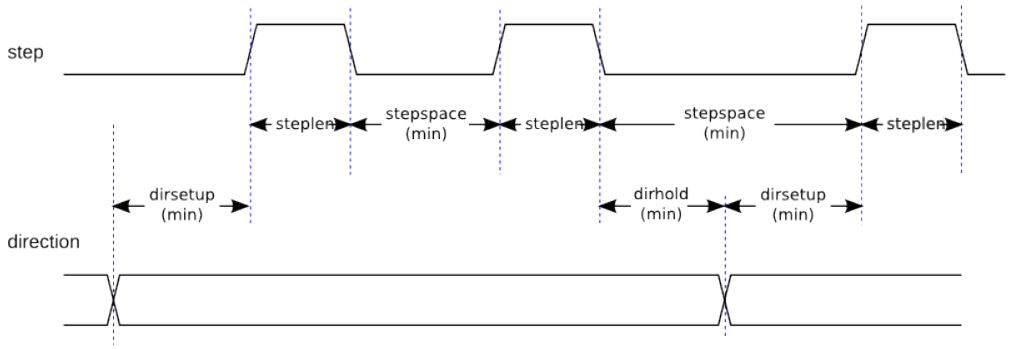


Figure 94. Schritt- und Richtungs-Timing (engl. step and direction timing)

Schritt Typ 1 (step type 1)

Der Schritttyp 1 hat zwei Ausgänge, aufwärts und abwärts. Die Impulse erscheinen je nach Fahrtrichtung an dem einen oder dem anderen. Jeder Impuls ist *steplen* ns lang, und die Impulse sind durch mindestens *stepspace* ns voneinander getrennt. Die maximale Frequenz ist dieselbe wie bei Schritttyp 0. Wenn *maxfreq* höher als der Grenzwert eingestellt ist, wird dieser gesenkt. Ist *maxfreq* gleich Null, bleibt er Null, aber die Ausgangsfrequenz wird trotzdem begrenzt.

WARNING Do not use the parport reset function with step types 2 - 15. Unexpected results can happen.

Step Type 2 - 15

Step types 2 through 15 are state based, and have from two to five outputs. On each step, a state counter is incremented or decremented. The Two-and-Three-Phase, Four-Phase, and Five-Phase show the output patterns as a function of the state counter. Step type 15 has no built-in pattern; its sequence is defined with the `user_step_type` module parameter (up to 18 state values, one per phase state). The maximum frequency is 1,000,000,000 divided by *steplen*, and as in the other modes, *maxfreq* will be lowered if it is above the limit.

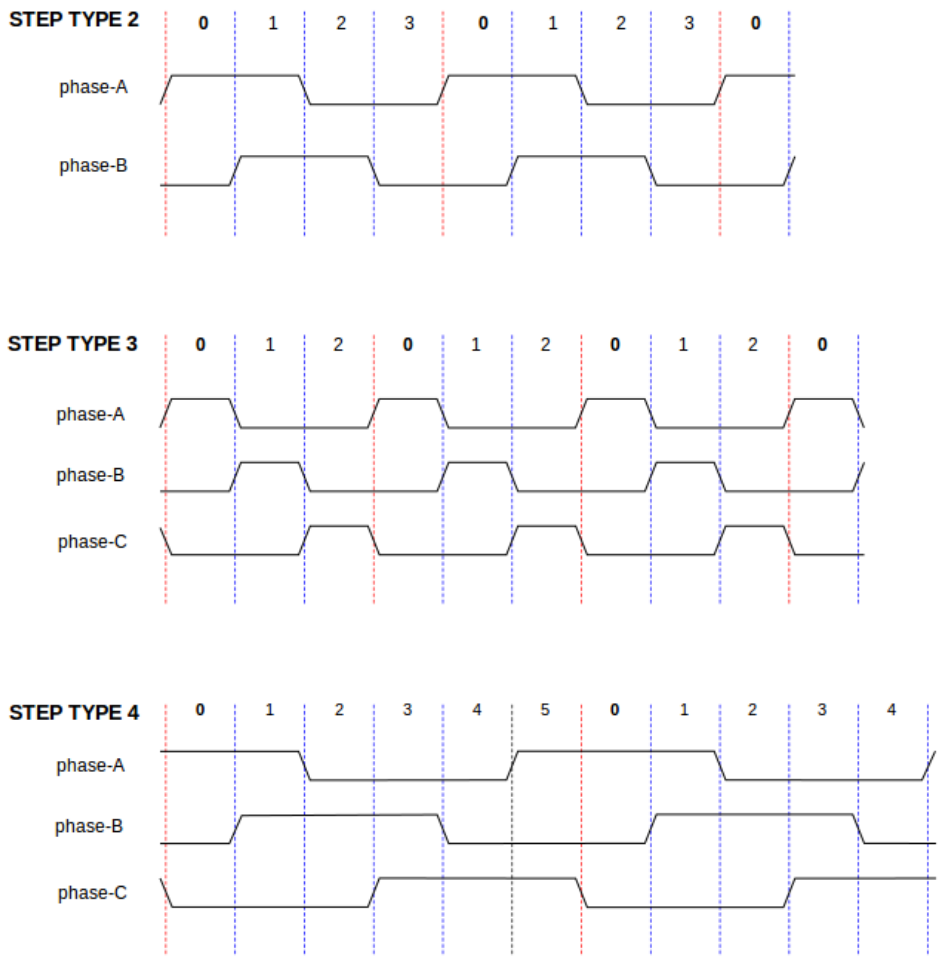


Figure 95. Zwei- und dreiphasige Schritttypen

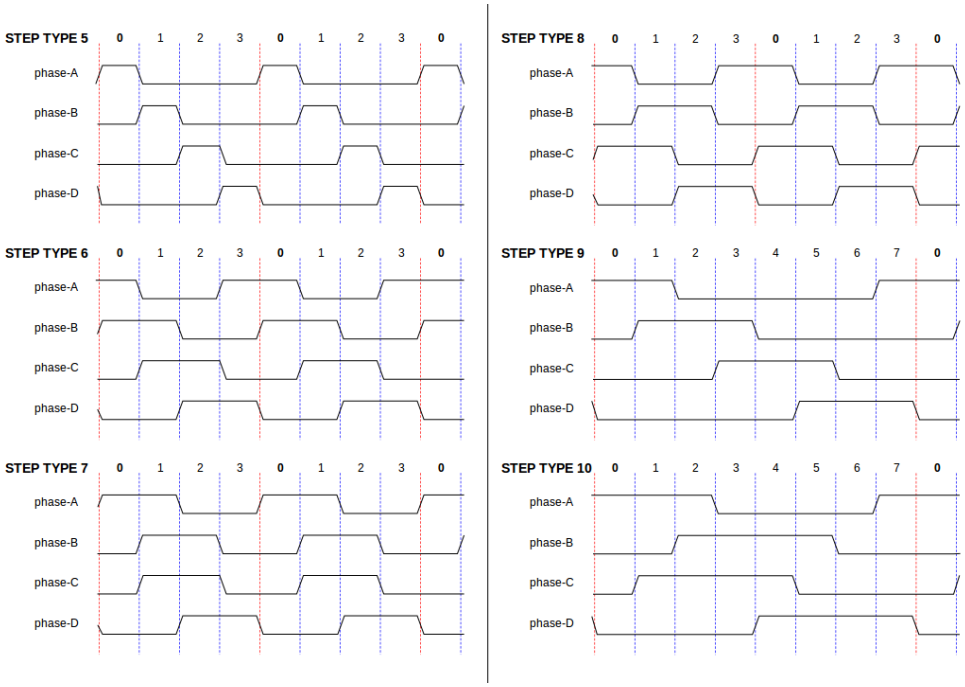


Figure 96. Vierphasige Schritttypen

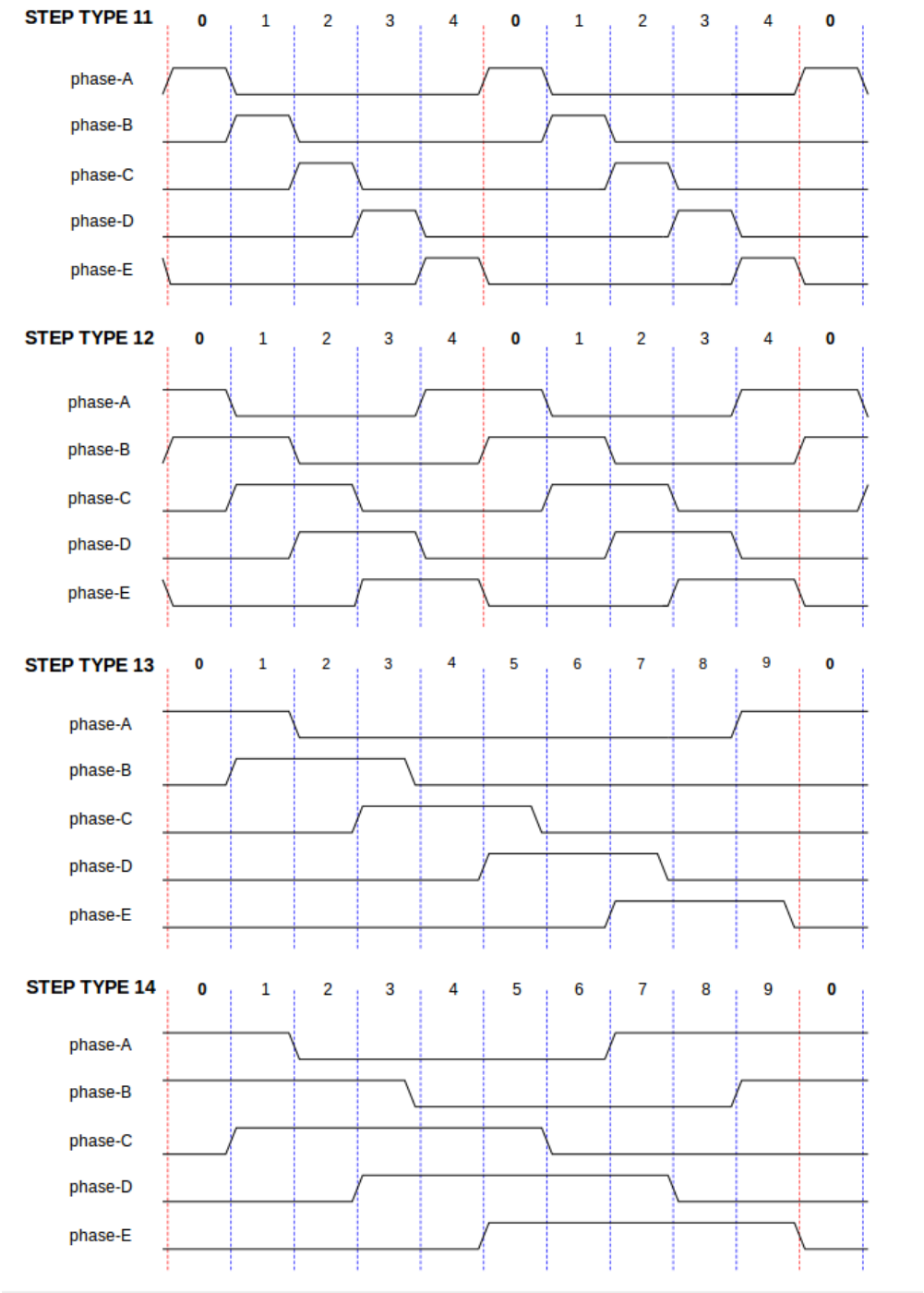


Figure 97. Fünf-Phasen-Schritttypen

Funktionen

Die Komponente exportiert drei Funktionen. Jede Funktion wirkt auf alle Schrittimpulsgeneratoren - die Ausführung verschiedener Generatoren in verschiedenen Threads wird nicht unterstützt.

- (funct) `stepgen.make-pulses` - High speed function to generate and count pulses.
- (funct) `stepgen.update-freq` - Die Funktion für niedrige Geschwindigkeiten wandelt Position in Geschwindigkeit um, skaliert und begrenzt.
- (funct) `stepgen.capture-position` - Funktion mit niedriger Geschwindigkeit für die Rückmeldung, aktualisiert die Zwischenspeicher und skaliert die Position.

Die Hochgeschwindigkeitsfunktion *stepgen.make-pulses* sollte in einem sehr schnellen Thread ausgeführt werden, je nach den Fähigkeiten des Computers zwischen 10 und 50 µs. Die Periode dieses Threads bestimmt die maximale Schrittfrequenz, da *steplen*, *stepspace*, *dirsetup*, *dirhold* und *dirdelay* alle auf ein ganzzahliges Vielfaches der Thread-Periode in Nanosekunden aufgerundet werden. Die beiden anderen Funktionen können mit einer viel geringeren Rate aufgerufen werden.

5.9.2. PWMgen

Diese Komponente ermöglicht die softwarebasierte Erzeugung von PWM- (Pulse Width Modulation) und PDM- (Pulse Density Modulation) Wellenformen. Es handelt sich um eine reine Echtzeitkomponente, die je nach CPU-Geschwindigkeit usw. PWM-Frequenzen von einigen hundert Hertz bei ziemlich guter Auflösung bis zu vielleicht 10 kHz mit begrenzter Auflösung erzeugen kann.

Laden von PWMgen

```
loadrt pwmgen output_type=<config-array>
```

Das <*config-array*> ist eine Reihe von durch Komma getrennten Dezimalzahlen. Jede Zahl bewirkt, dass ein einzelner PWM-Generator geladen wird; der Wert der Zahl bestimmt den Ausgangstyp. Im folgenden Beispiel werden drei PWM-Generatoren installiert. Es gibt keinen Standardwert, wenn <*config-array*> nicht angegeben wird, werden keine PWM-Generatoren installiert. Die maximale Anzahl von Frequenzgeneratoren ist 8 (wie durch MAX_CHAN in pwmgen.c definiert). Jeder Generator ist unabhängig, aber alle werden durch dieselbe(n) Funktion(en) zur gleichen Zeit aktualisiert. In den folgenden Beschreibungen steht <*chan*> für die Nummer eines bestimmten Generators. Der erste Generator hat die Nummer 0.

Beispiel für das Laden von PWMgen

```
loadrt pwmgen output_type=0,1,2
```

Es werden drei PWM-Generatoren installiert. Der erste wird einen Ausgang des Typs *0* (nur PWM) verwenden, der nächste einen Ausgang des Typs 1 (PWM und Richtung) und der dritte einen Ausgang des Typs 2 (AUF und AB). Es gibt keinen Standardwert, wenn <*config-array*> nicht angegeben wird, wird kein PWM-Generator installiert. Die maximale Anzahl von Frequenzgeneratoren ist 8 (wie durch MAX_CHAN in pwmgen.c definiert). Jeder Generator ist unabhängig, aber alle werden durch dieselbe(n) Funktion(en) zur gleichen Zeit aktualisiert. In den folgenden Beschreibungen steht <*chan*> für die Anzahl der einzelnen Generatoren. Die Nummerierung der PWM-Generatoren beginnt bei 0.

Entfernen (engl. hier unloading) von PWMgen

```
unloadrt pwmgen
```

Ausgangstypen (engl. output types)

Der PWM-Generator unterstützt drei verschiedene "Ausgangstypen".

- *Ausgangstyp 0* - Nur PWM-Ausgangspin. Nur positive Befehle werden akzeptiert, negative Werte werden als Null behandelt (und werden durch den Parameter *min-dc* beeinflusst, wenn er ungleich Null ist).

- *Ausgangstyp 1* - PWM/PDM und Richtungspins. Positive und negative Eingänge werden als positive und negative PWM ausgegeben. Der Richtungspin ist 0 für positive Befehle und 1 für negative Befehle. Wenn Ihre Steuerung positive PWM sowohl im Uhrzeigersinn als auch gegen den Uhrzeigersinn benötigt, verwenden Sie link:.. /man/man9/abs.9.html[abs]-Komponente, um Ihr PWM-Signal in einen positiven Wert umzuwandeln, wenn ein negativer Eingang eingegeben wird.
- *Ausgabetyp 2* - UP- und DOWN-Pins. Bei positiven Befehlen wird das PWM-Signal am Up-Ausgang angezeigt, und der Down-Ausgang bleibt false. Bei negativen Befehlen wird das PWM-Signal am Down-Ausgang angezeigt, und der Up-Ausgang bleibt false. Der Ausgangstyp 2 eignet sich für den Antrieb der meisten H-Brücken.

Pins

Jeder PWM-Generator hat die folgenden Pins:

- (real) `pwmgen.`__<chan>__.value` - Command value, in arbitrary units. Will be scaled by the *scale* parameter (see below).
- (bool) `pwmgen.`__<chan>__.enable` - Enables or disables the PWM generator outputs.

Jeder PWM-Generator verfügt über einige dieser Pins, je nach gewähltem Ausgangstyp:

- (bool) `pwmgen.`__<chan>__.pwm` - PWM (or PDM) output, (output types 0 and 1 only).
- (bool) `pwmgen.`__<chan>__.dir` - Direction output (output type 1 only).
- (bool) `pwmgen.`__<chan>__.up` - PWM/PDM output for positive input value (output type 2 only).
- (bool) `pwmgen.`__<chan>__.down` - PWM/PDM output for negative input value (output type 2 only).

Each PWM generator also has these pins:

- (real io) `pwmgen.`__<chan>__.scale` - Scaling factor to convert `value` from arbitrary units to duty cycle. For example if scale is set to 4000 and the input value passed to the `pwmgen.`__<chan>__.value` is 4000 then it will be 100% duty-cycle (always on). If the value is 2000 then it will be a 50% 25 Hz square wave.
- (real io) `pwmgen.`__<chan>__.offset` - DC offset added to `value` before scaling.
- (real io) `pwmgen.`__<chan>__.pwm-freq` - Desired PWM frequency, in Hz. If 0.0, generates PDM instead of PWM. If set higher than internal limits, next call of *pwmgen.update* will set it to the internal limit. If non-zero, and *dither* is false, next call of *pwmgen.update* will set it to the nearest integer multiple of the *make_pulses()* function period.
- (bool io) `pwmgen.`__<chan>__.dither-pwm` - If true, enables dithering to achieve average PWM frequencies or duty cycles that are unobtainable with pure PWM. If false, both the PWM frequency and the duty cycle will be rounded to values that can be achieved exactly.
- (real io) `pwmgen.`__<chan>__.min-dc` - Minimum duty cycle, between 0.0 and 1.0 (duty cycle will go to zero when disabled, regardless of this setting).
- (real io) `pwmgen.`__<chan>__.max-dc` - Maximum duty cycle, between 0.0 and 1.0.
- (real out) `pwmgen.`__<chan>__.curr-dc` - Current duty cycle - after all limiting and rounding (read

only).

Parameter

Keinen.

Funktionen

Die Komponente exportiert zwei Funktionen. Jede Funktion wirkt auf alle PWM-Generatoren - die Ausführung verschiedener Generatoren in verschiedenen Threads wird nicht unterstützt.

- (funct) `pwmgen.make-pulses` - High speed function to generate PWM waveforms. The high speed function `pwmgen.make-pulses` should be run in the base (fastest) thread, from 10 to 50 µs depending on the capabilities of the computer. That thread's period determines the maximum PWM carrier frequency, as well as the resolution of the PWM or PDM signals. If the base thread is 50,000 ns then every 50 µs the module decides if it is time to change the state of the output. At 50% duty cycle and 25 Hz PWM frequency this means that the output changes state every (1/25) s / 50 µs * 50% = 400 iterations. This also means that you have a 800 possible duty cycle values (without dithering).
- (funct) `pwmgen.update` - Funktion mit geringer Geschwindigkeit zur Skalierung und Begrenzung des Werts und zur Handhabung anderer Parameter. Dies ist die Funktion des Moduls, welche die komplizierteren mathematischen Berechnungen implementiert, um herauszufinden, für wie viele Basisperioden der Ausgang hoch und für wie viele er niedrig sein sollte.

5.9.3. Encoder

Diese Komponente ermöglicht die softwarebasierte Zählung von Signalen aus Quadratur- (oder Einzelimpuls-) Encodern. Es handelt sich um eine reine Echtzeitkomponente, die je nach CPU-Geschwindigkeit, Latenzzeit usw. maximale Zählraten von 10 kHz bis vielleicht 50 kHz erreichen kann.

Das Basisgewinde sollte 1/2 Zählgeschwindigkeit betragen, um Geräusche und Zeitschwankungen zu berücksichtigen. Wenn Sie z. B. einen Drehgeber mit 100 Impulsen pro Umdrehung an der Spindel haben und Ihre maximale Drehzahl 3000 beträgt, sollte das maximale Basisgewinde 25 µs betragen. Ein Drehgeber mit 100 Impulsen pro Umdrehung hat 400 Zählungen. Die Spindeldrehzahl von 3000 U/min (engl. RPM) = 50 U/s (engl. RPS, Umdrehungen pro Sekunde). 400 * 50 = 20.000 Zählungen pro Sekunde oder 50 µs zwischen den Zählungen.

Das Blockdiagramm des Encoderzählers ist ein Blockdiagramm eines Kanals eines Encoderzählers.

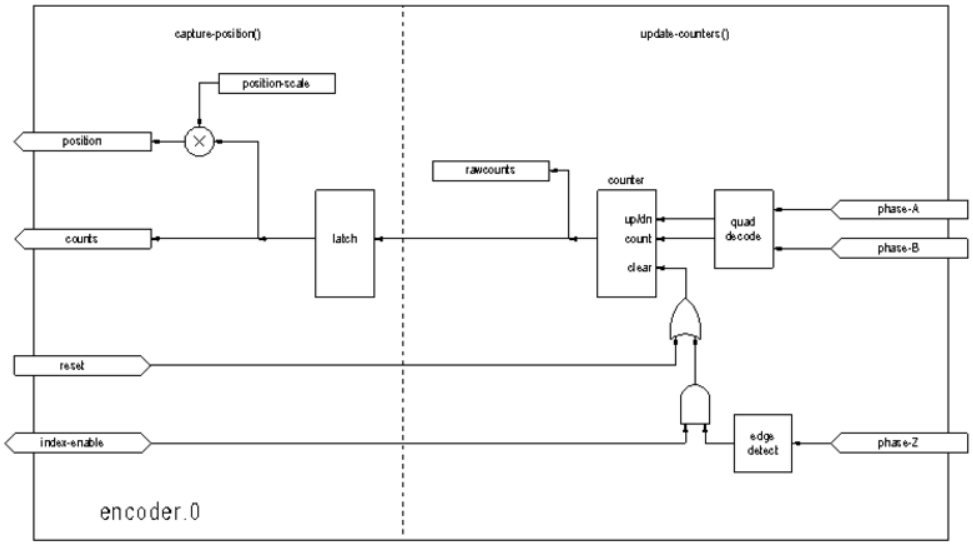


Figure 98. Blockdiagramm,Encoder Counter

Laden des Encoders

```
halcmd: loadrt encoder [num_chan=<counters>]
```

<*counters*> ist die Anzahl der Encoderzähler, die Sie installieren möchten. Wenn *num_chan* nicht angegeben ist, werden drei Zähler installiert. Die maximale Anzahl von Leistungsindikatoren beträgt 8 (wie durch MAX_CHAN in encoder.c definiert). Jeder Leistungsindikator ist unabhängig, aber alle werden gleichzeitig von den gleichen Funktionen aktualisiert. In den folgenden Beschreibungen ist <*chan*> die Nummer eines bestimmten Zählers. Der erste Zähler ist die Nummer 0.

Encoder entfernen (engl. unload)

```
halcmd: unloadrt encoder
```

Pins

- `encoder._<chan>_.counter-mode` (bool, i/o) (default: FALSE) - Enables counter mode. When true, the counter counts each rising edge of the phase-A input, ignoring the value on phase-B. This is useful for counting the output of a single channel (non-quadrature) sensor. When false, it counts in quadrature mode.
- `encoder._<chan>_.missing-teeth` (sint, in) (default: 0) - Enables the use of missing-tooth index. This allows a single IO pin to provide both position and index information. If the encoder wheel has 58 teeth with two missing, spaced as if there were 60(common for automotive crank sensors) then the position-scale should be set to 60 and missing-teeth to 2. To use this mode counter-mode should be set true. This mode will work for lathe threading but not for rigid tapping.
- `encoder._<chan>_.counts` (sint, out) - Position in encoder counts.

- encoder.<chan>.counts-latched (sint, out) - Not used at this time.
- encoder.<chan>.index-enable (bool, i/o) - When True, counts and position are reset to zero on next rising edge of Phase Z.
  At the same time, index-enable is reset to zero to indicate that the rising edge has occurred. The index-enable pin is bi-directional. If index-enable is False, the Phase Z channel of the encoder will be ignored, and the counter will count normally. The encoder driver will never set index-enable True. However, some other component may do so.
- encoder.<chan>.latch-falling (bool, in) (default: TRUE) - Not used at this time.
- encoder.<chan>.latch-input (bool, in) (default: FALSE) - Not used at this time.
- encoder.<chan>.latch-rising (bool, in) (default: TRUE) - Not used at this time.
- encoder.<chan>.min-speed-estimate (real, in) - Determine the minimum true velocity magnitude, at which velocity will be estimated as nonzero and position-interpolated will be interpolated. The units of min-speed-estimate are the same as the units of velocity. Scale factor, in counts per length unit. Setting this parameter too low will cause it to take a long time for velocity to go to 0 after encoder pulses have stopped arriving.
- encoder.<chan>.phase-A (bool, in) - Phase A of the quadrature encoder signal.
- encoder.<chan>.phase-B (bool, in) - Phase B of the quadrature encoder signal.
- encoder.<chan>.phase-Z (bool, in) - Phase Z (index pulse) of the quadrature encoder signal.
- encoder.<chan>.position (real, out) - Position in scaled units (see position-scale).
- encoder.<chan>.position-interpolated (real, out) - Position in scaled units, interpolated between encoder counts.
  The position-interpolated attempts to interpolate between encoder counts, based on the most recently measured velocity. Only valid when velocity is approximately constant and above min-speed-estimate. Do not use for position control, since its value is incorrect at low speeds, during direction reversals, and during speed changes.
  However, it allows a low ppr encoder (including a one pulse per revolution *encoder*) to be used for lathe threading, and may have other uses as well.
- encoder.<chan>.position-latched (real, out) - Not used at this time.
- encoder.<chan>.position-scale (real, i/o) - Scale factor, in counts per length unit. For example, if position-scale is 500, then 1000 counts of the encoder will be reported as a position of 2.0 units.
- encoder.<chan>.rawcounts (sint, out) - The raw count, as determined by update-counters. This value is updated more frequently than counts and position. It is also unaffected by reset or the index pulse.
- encoder.<chan>.reset (bool, in) - When True, force *counts* and *position* to zero immediately.
- encoder.<chan>.velocity (real, out) - Velocity in scaled units per second. encoder uses an algorithm that greatly reduces quantization noise as compared to simply differentiating the *position* output. When the magnitude of the true velocity is below min-speed-estimate, the velocity output is 0.
- encoder.<chan>.velocity-rpm (real, out) - Velocity in scaled units per minute.
- encoder.<chan>.x4-mode (bool, i/o) (default: TRUE) - Enables times-4 mode. When true, the

counter counts each edge of the quadrature waveform (four counts per full cycle). When false, it only counts once per full cycle. In counter-mode, this parameter is ignored. The 1x mode is useful for some jogwheels.

Parameter

Keinen.

Funktionen

Die Komponente exportiert zwei Funktionen. Jede Funktion wirkt auf alle Zähler des Encoders - die Ausführung verschiedener Zähler in verschiedenen Threads wird nicht unterstützt.

- (funct) `encoder.update-counters` - High speed function to count pulses.
- (funct) `encoder.capture-position` - Funktion mit niedriger Geschwindigkeit zur Aktualisierung von Latches und Skalenposition.

5.9.4. PID

Diese Komponente bietet Proportional/Integral/Derivativ-Regelkreise. Es handelt sich um eine reine Echtzeitkomponente. Der Einfachheit halber wird in dieser Diskussion davon ausgegangen, dass es sich um Positionsregelkreise handelt. Diese Komponente kann jedoch auch zur Implementierung anderer Rückkopplungsschleifen wie Geschwindigkeit, Brennerhöhe, Temperatur usw. verwendet werden. Das Blockdiagramm der PID-Schleife ist ein Blockdiagramm einer einzelnen PID-Schleife.

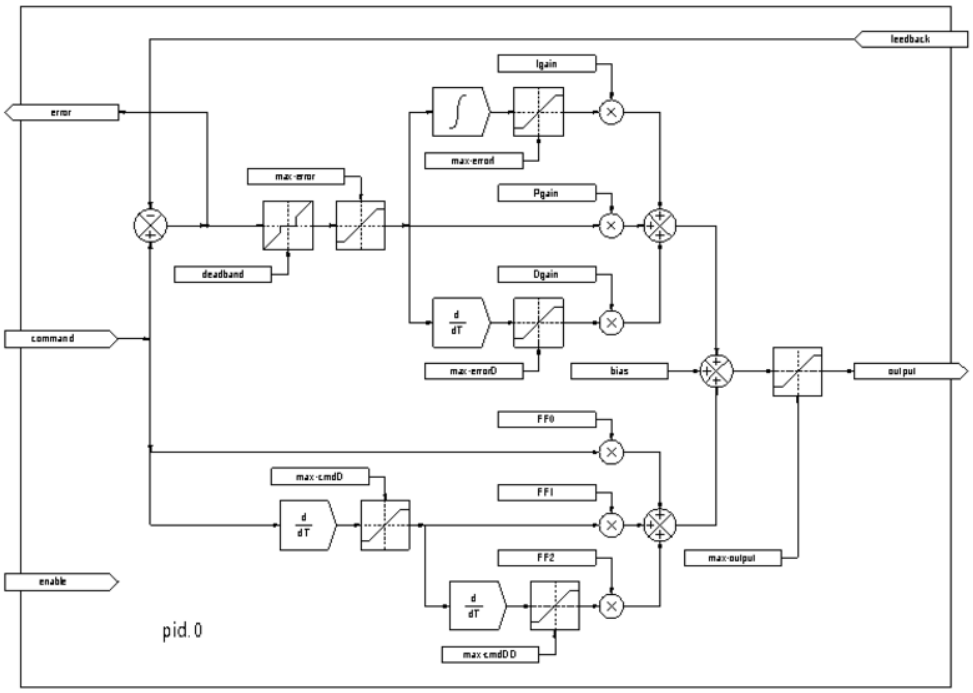


Figure 99. PID-Regelkreis-Blockdiagramm

PID laden

```
halcmd: loadrt pid [num_chan=<loops>] [debug=1]
```

<*Schleifen*> ist die Anzahl der PID-Schleifen, die Sie installieren möchten. Wird *num_chan* nicht angegeben, so wird eine Schleife installiert. Die maximale Anzahl von Schleifen ist 16 (wie durch MAX_CHAN in pid.c definiert). Jede Schleife ist völlig unabhängig. In den folgenden Beschreibungen ist <*Schleifennummer*> die Schleifennummer einer bestimmten Schleife. Die erste Schleife hat die Nummer 0.

Wenn `debug=1` angegeben ist, exportiert die Komponente einige zusätzliche Pins, die bei der Fehlersuche und beim Tuning nützlich sein können. Standardmäßig werden die zusätzlichen Pins nicht exportiert, um gemeinsamen Speicherplatz zu sparen und die Pin-Liste nicht zu überfrachten.

PID entfernen (engl. unload)

```
halcmd: unloadrt pid
```

Pins

Die drei wichtigsten Pins sind

- (real) `pid.`__<loopnum>__.command` - The desired position, as commanded by another system component.
- (real) `pid.`__<loopnum>__.feedback` - The present position, as measured by a feedback device such as an encoder.
- (real) `pid.`__<loopnum>__.output` - A velocity command that attempts to move from the present position to the desired position.

Bei einer Positionsschleife sind `.command` (engl. für Befehl) und `.feedback` (engl. für Rückmeldung) in Positionseinheiten angegeben. Bei einer linearen Achse können dies Zoll, mm, Meter oder andere relevante Einheiten sein. Bei einer Winkelachse kann es sich um Grad, Bogenmaß usw. handeln. Die Einheiten des `.output` Ausgangspins entsprechen der Änderung, die erforderlich ist, damit die Rückmeldung mit dem Befehl übereinstimmt. Bei einer Positionsschleife ist der "Ausgang" eine Geschwindigkeit in Zoll/Sekunde, mm/Sekunde, Grad/Sekunde usw. Zeiteinheiten sind immer Sekunden, und die Geschwindigkeitseinheiten entsprechen den Positionseinheiten. Wenn Befehl und Rückmeldung in Metern angegeben sind, erfolgt die Ausgabe in Metern pro Sekunde.

Jede Schleife hat zwei Pins, die zur Überwachung oder Steuerung des allgemeinen Betriebs der Komponente dienen.

- *(real) pid.<loopnum>.error* - Equals *.command* minus *.feedback*.
- *(bool) pid.<loopnum>.enable* - A bool that enables the loop. If *.enable* is false, all integrators are reset, and the output is forced to zero. If *.enable* is true, the loop operates normally.

Pins zur Meldung der Sättigung. Eine Sättigung ist gegeben, wenn der Ausgang des PID-Blocks an seinem maximalen oder minimalen Grenzwert liegt.

- *(bool) pid.<loopnum>.saturated* - True when output is saturated.
- *(real) pid.<loopnum>.saturated-s* - The time the output has been saturated.
- *(sint) pid.<loopnum>.saturated-count* - The time the output has been saturated.

Die PID-Verstärkungen, Grenzwerte und andere "abstimmbare" Merkmale des Regelkreises sind als Pins verfügbar, so dass sie dynamisch für erweiterte Abstimmungsmöglichkeiten angepasst werden können.

- *(real) pid.<loopnum>.Pgain* - Proportional gain
- *(real) pid.<loopnum>.Igain* - Integral gain
- *(real) pid.<loopnum>.Dgain* - Derivative gain
- *(real) pid.<loopnum>.bias* - Constant offset on output
- *(real) pid.<loopnum>.FF0* - Zeroth order feedforward - output proportional to command (position).
- *(real) pid.<loopnum>.FF1* - First order feedforward - output proportional to derivative of command (velocity).
- *(real) pid.<loopnum>.FF2* - Second order feedforward - output proportional to 2nd derivative of command (acceleration).
- *(real) pid.<loopnum>.FF3* - Third order feedforward - output proportional to 3rd derivative of command (jerk).
- *(real) pid.<loopnum>.deadband* - Amount of error that will be ignored
- *(real) pid.<loopnum>.maxerror* - Limit on error
- *(real) pid.<loopnum>.maxerrorI* - Limit on error integrator
- *(real) pid.<loopnum>.maxerrorD* - Limit on error derivative
- *(real) pid.<loopnum>.maxcmdD* - Limit on command derivative
- *(real) pid.<loopnum>.maxcmdDD* - Limit on command 2nd derivative
- *(real) pid.<loopnum>.maxcmdDDD* - Limit on command 3rd derivative
- *(real) pid.<loopnum>.maxoutput* - Limit on output value

Alle *max**-Grenzwerte sind so implementiert, dass es keinen Grenzwert gibt, wenn der Wert dieses Parameters Null ist.

Additional input pins:

- *(real) pid.<loopnum>.command-deriv* - Command derivative (velocity), used instead of differentiating *.command* when connected.
- *(real) pid.<loopnum>.feedback-deriv* - Feedback derivative (velocity), used instead of differentiating *.feedback* when connected.
- *(bool) pid.<loopnum>.index-enable* - When true, the loop is reset on the next rising edge, like an encoder index.
- *(bool) pid.<loopnum>.error-previous-target* - When true (the default), the derivative of error tracks the previous target to avoid derivative kick on command steps.

Auto-tuning pins (used by the *at_pid* component):

- *(real io) pid.<loopnum>.tune-effort* - Effort limit used during tuning cycles.
- *(uint io) pid.<loopnum>.tune-cycles* - Number of tuning cycles to run.
- *(uint io) pid.<loopnum>.tune-type* - Type of tuning to perform.
- *(bool) pid.<loopnum>.tune-mode* - When true, the loop runs in tuning mode.
- *(bool io) pid.<loopnum>.tune-start* - Set true to start tuning; reset to false when tuning is done.

If *debug=1* was specified when the component was installed, additional pins will be exported:

- *(real) pid.<loopnum>.errorI* - Integral of error.
- *(real) pid.<loopnum>.errorD* - Derivative of error.
- *(real) pid.<loopnum>.commandD* - Derivative of the command.
- *(real) pid.<loopnum>.commandDD* - 2nd derivative of the command.
- *(real) pid.<loopnum>.commandDDD* - 3rd derivative of the command.
- *(real) pid.<loopnum>.ultimate-gain* - Ultimate gain determined by auto-tuning.
- *(real) pid.<loopnum>.ultimate-period* - Ultimate period determined by auto-tuning.

Funktionen

Die Komponente exportiert eine Funktion für jede PID-Schleife. Diese Funktion führt alle für die Schleife erforderlichen Berechnungen durch. Da jede Schleife ihre eigene Funktion hat, können einzelne Schleifen in verschiedene Threads eingebunden werden und mit unterschiedlichen Geschwindigkeiten ausgeführt werden.

- *(funct) pid.<loopnum>.do-pid-calcs* - Performs all calculations for a single PID loop.

Wenn Sie den genauen Algorithmus zur Berechnung des Ausgangs der PID-Schleife verstehen möchten, lesen Sie bitte

- Abbildung PID Loop Block Diagram,
- die Kommentare am Anfang von *emc2/src/hal/components/pid.c*, und natürlich auf
- der G-Code selbst.

Die Schleifenberechnungen erfolgen in der C-Funktion „calc_pid()“.

5.9.5. Simulierter Encoder

Der simulierte Encoder ist genau das. Er erzeugt Quadraturimpulse mit einem Indeximpuls, und zwar mit einer durch einen HAL-Pin gesteuerten Geschwindigkeit. Er ist vor allem für Tests nützlich.

Sim-Encoder laden

```
halcmd: loadrt sim-encoder num_chan=<number>
```

<*number*> ist die Anzahl der Encoder, die Sie simulieren möchten. Wenn nicht angegeben, wird ein Encoder installiert. Die maximale Anzahl ist 8 (wie durch MAX_CHAN in sim_encoder.c definiert).

Abladen des sim-encoder

```
halcmd: unloadrt sim-encoder
```

Pins

- (real) `sim-encoder.`__<chan-num>__.speed` - The speed command for the simulated shaft.
- (bool) `sim-encoder.`__<chan-num>__.phase-A` - Quadrature output.
- (bool) `sim-encoder.`__<chan-num>__.phase-B` - Quadrature output.
- (bool) `sim-encoder.`__<chan-num>__.phase-Z` - Index pulse output.
- (sint) `sim-encoder.`__<chan-num>__.rawcounts` - Position in counts, can be driven to simulate an external encoder position.
- (uint io) `sim-encoder.`__<chan-num>__.ppr` - Pulses Per Revolution.
- (real io) `sim-encoder.`__<chan-num>__.scale` - Scale Factor for `.speed`. The default is 1.0, which means that `.speed` is in revolutions per second. Change to 60 for RPM, to 360 for degrees per second, 6.283185 (= 2*π) for radians per second, etc.

Wenn `.speed` positiv ist, liegt `.phase-A` vor `.phase-B`.

Parameter

Keinen.

Beachten Sie, dass Impulse pro Umdrehung nicht dasselbe sind wie Zählungen pro Umdrehung. Ein Impuls ist ein vollständiger Quadraturzyklus. Die meisten Drehgeberzähler zählen viermal während eines vollständigen Zyklus.

Funktionen

Die Komponente exportiert zwei Funktionen. Jede Funktion wirkt auf alle simulierten Geber.

- (funct) `sim-encoder.make-pulses` - High speed function to generate quadrature pulses.
- (funct) `sim-encoder.update-speed` - Funktion für niedrige Geschwindigkeit zum Lesen von `.speed`, Skalieren und Einrichten von `.make-pulses`.

5.9.6. Debounce

Die Entprellung ist eine Echtzeitkomponente, zum Herausfiltern der durch mechanische Schaltkontakte verursachten Störungen. Sie kann auch in anderen Anwendungen nützlich sein, in denen kurze Impulse unterdrückt werden müssen.

Debounce wird geladen

```
halcmd: loadrt debounce cfg=<config-string>
```

<Konfigurations-Zeichenfolge>

Ist eine Reihe von durch Komma getrennten Dezimalzahlen. Jede Zahl installiert eine Gruppe identischer Entprellungsfilter, wobei die Zahl angibt, wie viele Filter in der Gruppe enthalten sind.

Beispiel zum Laden von Debounce

```
halcmd: loadrt debounce cfg=1,4,2
```

will install three groups of filters. Group 0 contains one filter, group 1 contains four, and group 2 contains two filters. There is no default value for <*config-string*>; at least one group size must be given or loading fails. The maximum number of groups is 8 (as defined by MAX_GROUP in debounce.c). The maximum number of filters in a group is 50 (as defined by MAX_GROUP_SIZE in debounce.c). Each group is completely independent. All filters in a single group are identical, and they are all updated by the same function at the same time. In the following descriptions, <*G*> is the group number and <*F*> is the filter number within the group. The first filter is group 0, filter 0.

Entladen der Entprellung

```
halcmd: unloadrt debounce
```

Pins

Jeder einzelne Filter hat zwei Pins.

- (bool) `debounce.`__<G>__.__<F>__.in` - Input of filter <*F*> in group <*G*>.
- (bool) `debounce.`__<G>__.__<F>__.out` - Output of filter <*F*> in group <*G*>.

Parameter

Jede Gruppe von Filtern hat einen ParameterFußnote:[Jeder einzelne Filter hat auch eine interne Statusvariable. Es gibt einen Kompilierzeitschalter, der diese Variable als Parameter exportieren kann. Dies ist für Tests gedacht und verschwendet unter normalen Umständen nur gemeinsamen Speicher.].

- (sint) `debounce.`__<G>__.delay` - Filter delay for all filters in group <*G*>.

Die Filter-Verzögerung (engl. delay) wird in Einheiten von Thread-Perioden angegeben. Die minimale Verzögerung ist Null. Der Ausgang eines Filters mit einer Verzögerung von Null folgt genau seinem Eingang - er filtert nichts. Mit zunehmendem `.delay` werden immer längere Störimpulse zurückgewiesen. Wenn `.delay` 4 ist, werden alle Störungen zurückgewiesen, die kleiner oder gleich vier Thread-Perioden sind.

Funktionen

Jede Gruppe von Filtern hat eine Funktion, die alle Filter in dieser Gruppe "gleichzeitig" aktualisiert.

Verschiedene Gruppen von Filtern können von verschiedenen Threads in verschiedenen Zeiträumen aktualisiert werden.

- (funct) `debounce.`<*G*> - Aktualisiert alle Filter in der Gruppe <*G*>.

5.9.7. SigGen

SigGen ist eine Echtzeitkomponente, die Rechteck-, Dreieck- und Sinuswellen erzeugt. Sie wird hauptsächlich zum Testen verwendet.

Laden von siggen

```
halcmd: loadrt siggen [num_chan=<chans>]
```

<chans>

ist die Anzahl der Signalgeber, die Sie installieren möchten. Wenn *numchan* nicht angegeben wird, dann wird ein Signalgenerator installiert. Die maximale Anzahl von Generatoren ist 16 (wie durch MAX_CHAN in siggen.c definiert). Jeder Generator ist völlig unabhängig. In den folgenden Beschreibungen ist

<*chan*>

die Nummer eines bestimmten Signalgebers (die Nummern beginnen bei 0).

Entladen (engl. unload) von Siggen

```
halcmd: unloadrt siggen
```

Pins

Jeder Generator hat fünf Ausgangspins.

- (real) siggen.`__<chan>__.sine` - Sine wave output.
- (real) siggen.`__<chan>__.cosine` - Cosine output.
- (real) siggen.`__<chan>__.sawtooth` - Sawtooth output.
- (real) siggen.`__<chan>__.triangle` - Triangle wave output.
- (real) siggen.`__<chan>__.square` - Square wave output.
- (bool) siggen.`__<chan>__.clock` - Clock output, toggles once per cycle.

Alle fünf Ausgänge haben die gleiche Frequenz, Amplitude und Offset.

Zusätzlich zu den Ausgangspins gibt es drei Steuerpins:

- (real) siggen.`__<chan>__.frequency` - Sets the frequency in Hertz, default value is 1 Hz.
- (real) siggen.`__<chan>__.amplitude` - Sets the peak amplitude of the output waveforms, default is 1.
- (real) siggen.`__<chan>__.offset` - Sets DC offset of the output waveforms, default is 0.

- (bool) siggen.`__<chan>__.reset` - When true, the internal phase accumulator is held at zero.

Wenn zum Beispiel siggen.0.amplitude 1,0 und siggen.0.offset 0,0 ist, schwanken die Ausgänge von -1,0 bis +1,0. Wenn siggen.0.amplitude 2,5 und siggen.0.offset 10,0 ist, schwanken die Ausgänge zwischen 7,5 und 12,5.

Parameter

Keine. [5]

Funktionen

- (funct) siggen.`__<chan>__.update` - Berechnet neue Werte für alle fünf Ausgaben.

5.9.8. lut5

Die Komponente lut5 ist eine Logikkomponente mit 5 Eingängen, die auf einer Look-up-Tabelle basiert.

- lut5 does not use floating point math.

Laden von lut5

```
loadrt lut5 [count=N|names=name1[,name2...]]
addf lut5.N servo-thread | base-thread
setp lut5.N.function 0xN
```

lut5*-Rechenfunktion*

Um die hexadezimale Zahl für die Funktion zu berechnen, fangen Sie oben an und schreiben Sie eine 1 oder 0, um anzugeben, ob diese Zeile wahr oder falsch ist. Als Nächstes notieren Sie jede Zahl in der Ausgabespalte, beginnend von oben und von rechts nach links. Dies wird die Binärzahl sein. Mit einem Taschenrechner mit einer Programmansicht wie der in Ubuntu geben Sie die Binärzahl ein und konvertieren sie dann in Hexadezimal und das ist dann der Wert für die Funktion.

Table 11. lut5 *Look Up Table*

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Ausgabe
0	0	0	0	0	
0	0	0	0	1	
0	0	0	1	0	
0	0	0	1	1	
0	0	1	0	0	
0	0	1	0	1	
0	0	1	1	0	

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Ausgabe
0	0	1	1	1	
0	1	0	0	0	
0	1	0	0	1	
0	1	0	1	0	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	0	1	
0	1	1	1	0	
0	1	1	1	1	
1	0	0	0	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	0	1	1	
1	0	1	0	0	
1	0	1	0	1	
1	0	1	1	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	0	1	
1	1	0	1	0	
1	1	0	1	1	
1	1	1	0	0	
1	1	1	0	1	
1	1	1	1	0	
1	1	1	1	1	

`lut5` *Zwei Eingänge Beispiel*

In der folgenden Tabelle haben wir für jede Zeile den Ausgangszustand ausgewählt, den wir für wahr halten wollen.

Table 12. `lut5` *Zwei Eingänge Beispiel Look Up Table*

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Ausgabe
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	1

In der Ausgangsspalte unseres Beispiels soll der Ausgang eingeschaltet sein, wenn Bit 0 oder Bit 0 und Bit1 eingeschaltet sind und sonst nichts. Die binäre Zahl ist *b1010* (drehen Sie den Ausgang um 90 Grad nach rechts). Geben Sie diese Zahl in den Taschenrechner ein und stellen Sie die Anzeige auf hexadezimal um. Das hexadezimale Präfix ist *0x*.

5.10. HAL Komponenten Erstellung

5.10.1. Einführung

In diesem Abschnitt wird die Zusammenstellung von HAL-Komponenten vorgestellt, d. h. die Hinzufügung einiger Kenntnisse der Maschinenbediener über den Umgang mit der Maschine. Es ist zu beachten, dass solche Komponenten nicht unbedingt direkt mit der Hardware zu tun haben. Sie tun es oft, aber nicht notwendigerweise, z.B. könnte es eine Komponente geben, die zwischen imperialen und metrischen Maßstäben umrechnet, so dass es in diesem Abschnitt nicht erforderlich ist, auf die Interaktion mit der Hardware einzugehen.

Das Schreiben einer HAL-Komponente kann ein langwieriger Prozess sein, die meisten davon in Setup-Aufrufe zu *rtapi_* und *hal_* Funktionen und damit verbundene Fehlerprüfung. *halcompile* wird all diesen Code für Sie schreiben, automatisch. Das Kompilieren einer HAL-Komponente ist auch viel einfacher, wenn man *halcompile* benutzt, egal ob die Komponente Teil des LinuxCNC-Source-Trees ist, oder außerhalb davon.

Eine einfache Komponente wie "ddt", die in C kodiert ist, umfasst beispielsweise etwa 80 Zeilen Code. Die entsprechende Komponente ist sehr kurz, wenn sie mit dem Präprozessor "halcompile" geschrieben wird:

Beispiel für eine einfache Komponente

```
component ddt "Compute the derivative of the input function";
pin in real in;
pin out real out;
variable rtapi_real old;
function _;
license "GPL"; // indicates GPL v2 or later
;;
rtapi_real tmp = in;
out_set((tmp - old) / fperiod);
old = tmp;
```

5.10.2. Installation

Um eine Komponente zu kompilieren, wenn eine gepackte Version von LinuxCNC verwendet wird, müssen Entwicklungspakete installiert werden, indem man entweder Synaptic aus dem Hauptmenü *System → Administration → Synaptic package manager* benutzt oder einen der folgenden Befehle in einem Terminalfenster ausführt:

Installation von Entwicklungspaketen für LinuxCNC

```
sudo apt install linuxcnc-dev
# oder
sudo apt install linuxcnc-uspace-dev
```

Eine andere Methode ist die Verwendung des Synaptic-Paketmanagers aus dem Anwendungsmenü, um die Pakete `linuxcnc-dev` oder `linuxcnc-uspace-dev` zu installieren.

5.10.3. Kompilieren

Mittendrin im Quellcode

Legen Sie die `.comp`-Datei in das Quellverzeichnis `linuxcnc/src/hal/components` und führen Sie `make` erneut aus. *Comp*-Dateien werden vom Build-System automatisch erkannt.

Wenn eine `.comp`-Datei ein Treiber für Hardware ist, kann sie in `linuxcnc/src/hal/drivers` abgelegt werden und wird gebaut, es sei denn, LinuxCNC ist als Nicht-Echtzeit-Simulator konfiguriert.

Echtzeit-Komponenten außerhalb des Quellbaums

halcompile kann eine Echtzeitkomponente in einem einzigen Schritt verarbeiten, kompilieren und installieren, wobei `rtexample.ko` im LinuxCNC-Echtzeitmodulverzeichnis platziert wird:

```
[sudo] halcompile --install rtexample.comp
```

NOTE sudo (für Root-Rechte) wird benötigt, wenn Sie LinuxCNC aus einem Deb-Paket installieren. Wenn Sie einen Run-In-Place (RIP) Build verwenden, sollten Root-Rechte nicht erforderlich sein.

Or, it can process and compile in one step, leaving `example.ko` (or `example.so` for uspace) in the current directory:

```
halcompile --compile rtexample.comp
```

Oder es kann einfach verarbeitet werden, wobei die Datei `example.c` im aktuellen Verzeichnis verbleibt:

```
halcompile rtexample.comp
```

halcompile kann auch eine in C geschriebene Komponente kompilieren und installieren, indem es die

oben gezeigten Optionen `--install` und `--compile` verwendet:

```
[sudo] halcompile --install rtexample2.c
```

Die Dokumentation im man-Format kann auch aus den Informationen im Deklarationsabschnitt erstellt werden:

```
halcompile --document -o example.9 rtexample.comp
```

Die resultierende Manpage „example.9“ kann angezeigt werden mit

```
man ./example.9
```

oder an einen Standardspeicherort für UNIX man pages kopiert.

Kompilieren von Nicht-Echtzeitkomponenten außerhalb des Quellbaums

halcompile kann Nicht-Echtzeit-Komponenten verarbeiten, kompilieren, installieren und dokumentieren:

```
halcompile non-rt-example.comp
halcompile --compile non-rt-example.comp
[sudo] halcompile --install non-rt-example.comp
halcompile --document non-rt-example.comp
```

Bei einigen Bibliotheken (z. B. Modbus) kann es erforderlich sein, zusätzliche Compiler- und Linker-Argumente hinzuzufügen, damit der Compiler die Bibliotheken finden und linken kann. Im Falle von `.comp`-Dateien kann dies über "option"-Anweisungen in der .comp-Datei erfolgen. Für `.c`-Dateien ist dies nicht möglich, so dass stattdessen die Parameter `--extra-compile-args` und `--extra-link-args` verwendet werden können. Als Beispiel kann diese Befehlszeile verwendet werden, um die Komponente `vfdb_vfd.c` out-of-tree zu kompilieren.

```
halcompile --userspace --install --extra-compile-args="-I/usr/include/modbus" --extra
-link-args="-lm -lmodbus -llinuxcncini" vfdb_vfd.c
```

NOTE Die Auswirkung der Verwendung von extra-args in der Befehlszeile und in der Datei ist undefiniert.

5.10.4. Verwendung einer Komponente

Die Komponenten müssen geladen und zu einem Thread hinzugefügt werden, bevor sie eingesetzt werden können. Die bereitgestellte Funktionalität kann dann direkt und wiederholt von einem der Threads aufgerufen werden oder sie wird von anderen Komponenten aufgerufen, die ihre eigenen Auslöser haben.

Beispiel für ein HAL-Skript zur Installation einer Komponente (ddt), die jede Millisekunde ausgeführt wird.

```
loadrt threads name1=servo-thread period1=1000000
loadrt ddt
addf ddt.0 servo-thread
```

Weitere Informationen zu `loadrt` and `addf` sind bei den HAL Grundlagen zu finden.

Um Ihre Komponente zu testen, können Sie den Beispielen im HAL Tutorial folgen.

5.10.5. Definitionen

- **component** - Eine Komponente (engl. component) ist ein einzelnes Echtzeitmodul, das mit `halcmd loadrt` geladen wird. Eine *.comp*-Datei gibt eine Komponente an. Der Komponentenname und der Dateiname müssen übereinstimmen.
- **instance** – Eine Komponente kann null oder mehr Instanzen (engl. instance) haben. Jede Instanz einer Komponente wird gleich erstellt (sie haben alle dieselben Pins, Parameter, Funktionen und Daten), verhalten sich jedoch unabhängig, wenn ihre Pins, Parameter und Daten unterschiedliche Werte haben.
- **singleton** - Es ist möglich, dass eine Komponente ein "Singleton" ist, in diesem Fall wird genau eine Instanz erstellt. Es ist selten sinnvoll, eine "Singleton"-Komponente zu schreiben, es sei denn, es kann buchstäblich nur ein einziges Objekt dieser Art im System geben (z.B. eine Komponente, deren Zweck es ist, einen Pin mit der aktuellen UNIX-Zeit zu versehen, oder ein Hardware-Treiber für den internen PC-Lautsprecher).

5.10.6. Erstellung einer Instanz

Bei einem Singleton wird eine Instanz erstellt, wenn die Komponente geladen wird.

Bei einem Nicht-Singleton bestimmt der Modulparameter "count", wie viele nummerierte Instanzen erstellt werden. Wenn *count* nicht angegeben wird, bestimmt der Modulparameter *names*, wie viele benannte Instanzen erstellt werden. Wenn weder *count* noch *names* angegeben werden, wird eine einzige nummerierte Instanz erstellt.

5.10.7. Implizite Parameter

Den Funktionen wird implizit der Parameter *period* übergeben, der die Zeit in Nanosekunden der letzten Periode zur Ausführung der Komponente angibt. Funktionen, die Fließkommazahlen verwenden, können sich auch auf den Parameter *fperiod* beziehen, der die Fließkommazeit in Sekunden oder (period*1e-9) angibt. Dies kann in Komponenten nützlich sein, die Zeitinformationen benötigen. Siehe auch die nachfolgend beschriebene Option *period*.

5.10.8. Syntax

Eine `.comp'`-Datei besteht aus einer Reihe von Deklarationen, gefolgt von `;;` auf einer eigenen Zeile, gefolgt von C Code, der die Funktionen des Moduls implementiert.

Die Erklärungen umfassen:

- *component HALNAME (DOC);*
- *pin PINDIRECTION TYPE HALNAME ([SIZE]|[MAXSIZE: CONDSIZE]) (if CONDITION) (= STARTVALUE) (DOC) ;*
- *param PARAMDIRECTION TYPE HALNAME ([SIZE]|[MAXSIZE: CONDSIZE]) (if CONDITION) (= STARTVALUE) (DOC) ;*
- *function HALNAME (fp | nofp) (DOC);*
- *option OPT (VALUE);*
- *variable CTYPE STARREDNAME ([SIZE]);*
- *description DOC;*
- *examples DOC;*
- *notes DOC;*
- see_also DOC;'
- *license LICENSE;*
- *author AUTHOR;*
- *include HEADERFILE;*

Klammern kennzeichnen optionale Elemente. Ein senkrechter Strich kennzeichnet Alternativen. Wörter in "GROSSBUCHSTABEN" kennzeichnen variablen Text, wie folgt:

- *NAME* - Ein Standard-C-Bezeichner
- STARREDNAME' - Ein C-Bezeichner mit null oder mehr * vor dem Namen. Diese Syntax kann verwendet werden, um Instanzvariablen zu deklarieren, die Zeiger sind. Beachten Sie, dass aufgrund der Grammatik kein Leerzeichen zwischen dem * und dem Variablennamen stehen darf.
- *HALNAME* - Ein erweiterter Bezeichner. Bei der Erstellung eines HAL-Bezeichners werden alle Unterstriche durch Bindestriche ersetzt, und alle nachgestellten Bindestriche oder Punkte werden entfernt, so dass "this_name_" in "dieser-Name" umgewandelt wird, und wenn der Name "_" ist, wird auch ein nachgestellter Punkt entfernt, so dass "function _" einen HAL-Funktionsnamen wie "component" ergibt. " <num>statt "Komponente. <num>."

  Falls vorhanden, wird beim Erstellen von Pins, Parametern und Funktionen das Präfix *hal_* am Anfang des Komponentennamens entfernt.

Im HAL-Bezeichner für einen Pin oder Parameter kennzeichnet # ein Arrayelement und muss in Verbindung mit einer *[SIZE]*-Deklaration verwendet werden. Die Rautenzeichen werden durch eine 0-aufgefüllte Zahl ersetzt mit der gleichen Länge wie die Anzahl der #-Zeichen.

Wenn Sie einen C-Bezeichner erstellen, werden die folgenden Änderungen am HALNAME vorgenommen:

1. Alle "#"-Zeichen und alle Zeichen ".", "_" oder "-", die unmittelbar davor stehen, werden entfernt.
2. Alle verbleibenden "."- und "-"-Zeichen werden durch "_" ersetzt.

3. Wiederholte „_“-Zeichen werden in ein einzelnes „_“-Zeichen geändert.

Ein nachgestelltes "_" wird beibehalten, damit HAL-Kennungen, die sonst mit reservierten Namen oder Schlüsselwörtern (z. B. "min") kollidieren würden, verwendet werden können.

HALNAME	C Bezeichner (engl. identifier)	HAL-Bezeichner (engl. identifier)
x_y_z	x_y_z	x-y-z
x-y.z	x_y_z	x-y.z
x_y_z_	x_y_z_	x-y-z
x.##.y	x_y(MM)	x.MM.z
x.##	x(MM)	x.MM

NOTE

Two declarations that claim the same HAL identifier — *x_y_z* and *x_y_z_* in the table above — are rejected by `halcompile`; they would otherwise be refused by HAL at `loadrt`. An array claims one identifier per element, so *x_#* with *[4]* and *x_0* collide, and a function claims *<name>.time*, *<name>.tmax* and *<name>.tmax-increased* as well, since those are created alongside it. Pins and parameters share one namespace; functions have their own. An *if* condition does not exempt a declaration.

- *if CONDITION* (engl. für Bedingung)- Ein Ausdruck mit der Variablen *Persönlichkeit*, die ungleich Null ist, wenn der Pin oder Parameter erstellt werden soll.
- *SIZE* - A number that gives the size of an array, at most 256. The array items are numbered from 0 to *SIZE*-1.
- *MAXSIZE : CONDSIZE* - A number that gives the maximum size of the array, at most 256, followed by an expression involving the variable *personality* and which always evaluates to less than *MAXSIZE*. When the array is created its size will be *CONDSIZE*.
- *DOC* - Eine Zeichenfolge, die das Element dokumentiert. Die Zeichenfolge kann eine "doppelt in Anführungszeichen" gesetzte Zeichenfolge im C-Stil sein, z. B.:

  ```
  "Wählt die gewünschte Flanke aus: TRUE bedeutet fallend, FALSE bedeutet steigend"
  ```

  oder eine "dreifach in Anführungszeichen" gesetzte Zeichenfolge im Python-Stil, die eingebettete Zeilenumbrüche und Anführungszeichen enthalten kann, z. B.:

  ```
  """Die Wirkung dieses Parameters, auch bekannt als "der Orb von Zot",
  ist in mindestens zwei Absätzen zu erklären.

  Hoffentlich haben Ihnen diese Absätze geholfen, "zot" besser
  zu verstehen."""
  ```

  Einer Zeichenkette kann auch das Literalzeichen *r* vorangestellt werden; in diesem Fall wird die Zeichenkette wie eine Python-Rohzeichenkette interpretiert.

The documentation string is in "asciidoc" format. For more information on this markup format, see *ascidoctor(1)* and https://asciidoctor.org.

- *TYPE* - One of the HAL types: `bool`, `sint`, `uint` or `real`.
- *PINDIRECTION* - One of the following: `in`, `out`, or `io`. A component sets a value for an *out* pin, it reads a value from an `in` pin, and it may read or set the value of an `io` pin.
- *PARAMDIRECTION* - One of the following: `r` or `rw`. A component sets a value for a *r* parameter, and it may read or set the value of a *rw* parameter.
- *STARTVALUE* - Specifies the initial value of a pin or parameter. If it is not specified, then the default is `0` or `FALSE`, depending on the type of the item.
- *HEADERFILE* - Der Name einer Headerdatei, entweder in doppelten Anführungszeichen (*include "myfile.h";*) oder in spitzen Klammern (*include <systemfile.h>;*). Die Header-Datei wird (unter Verwendung der #include von C) am Anfang der Datei vor Pin- und Parameterdeklarationen eingefügt.

NOTE

Transitionally, `si32` and `ui32` are supported as pin and param types. These read the lower 32-bit value (truncating!) and write 32-bit (properly sign-extended for the underlying type). The underlying values are always correct.

You should retire all 32-bit pins and params and upgrade to *sint* and *uint*. Please beware of local variable truncation in the upgrade process. The proper variable types matching for all HAL types are:

HAL type	C type	Beschreibung
`bool`	`rtapi_bool`	Boolean value
`real`	`rtapi_real`	Floating point value
`sint`	`rtapi_sint`	Signed integer value
`uint`	`rtapi_uint`	Unsigned integer value

Optionen

Die derzeit definierten Optionen sind:

- *option singleton yes* - (Voreinstellung: no)
  Erzeugt keinen *count*-Modulparameter und immer eine einzelne Instanz. Mit *singleton* werden die Elemente *Komponentenname.Elementname* genannt und ohne *singleton* werden die Elemente für nummerierte Instanzen *Komponentenname.<num>.Elementname* genannt.
- option default_count number' - (Standardwert: 1)
  Normalerweise ist der Modulparameter *count* auf 1 voreingestellt. Ist er angegeben, so wird *count* stattdessen auf diesen Wert gesetzt.
- *option count_function yes* - (default: no)
  Normally, the number of instances to create is specified in the module parameter *count*; if *count_function* is specified, the value returned by the function *int get_count(void)* is used instead, and

the *count* module parameter is not defined. In userspace components, *names*= is not supported with *count_function*.

- *option rtapi_app no* - (Voreinstellung: yes)
  Normalerweise werden die Funktionen `rtapi_app_main()` und `rtapi_app_exit()` automatisch definiert. Bei *option rtapi_app no* sind sie es nicht und müssen im C-Code bereitgestellt werden. Verwenden Sie die folgenden Prototypen:

  ```
  `int rtapi_app_main(void);`

  `void rtapi_app_exit(void);`
  ```

  Wenn Sie Ihre eigene `rtapi_app_main()` implementieren, rufen Sie die Funktion `int export(char *prefix, long extra_arg)` auf, um die Pins, Parameter und Funktionen für `prefix` zu registrieren.

- *option data TYPE* - (default: none) **deprecated**
  If specified, each instance of the component will have an associated data block of type *TYPE* (which can be a simple type like *real* or the name of a type created with *typedef*). In new components, *variable* should be used instead.

- *option extra_setup yes* - (Voreinstellung: no)
  Wenn angegeben, wird die durch *EXTRA_SETUP* definierte Funktion für jede Instanz aufgerufen. Bei Verwendung der automatisch definierten *rtapi_app_main* ist *extra_arg* die Nummer dieser Instanz.

- option extra_cleanup yes' - (Voreinstellung: no)
  Wenn angegeben, wird die durch *EXTRA_CLEANUP* definierte Funktion aus dem automatisch definierten *rtapi_app_exit* oder, im Falle eines erkannten Fehlers, im automatisch definierten *rtapi_app_main* aufgerufen.

- *option post_export yes* - (default: no)
  If specified, call the function defined by *POST_EXPORT* for each instance. If using the automatically defined *rtapi_app_main*, *extra_arg* is the number of this instance.

- *option userspace yes* - (Voreinstellung: no)
  Falls angegeben, beschreibt diese Datei eine Nicht-Echtzeit-Komponente (früher bekannt als "Userspace") und nicht eine reguläre (d.h. Echtzeit-) Komponente. Eine Nicht-Echtzeit-Komponente kann keine Funktionen haben, die durch die *function*-Direktive definiert sind. Stattdessen wird, nachdem alle Instanzen konstruiert sind, die C-Funktion `void user_mainloop(void);` aufgerufen. Wenn diese Funktion zurückkehrt, wird die Komponente beendet. Normalerweise verwendet *user_mainloop() FOR_ALL_INSTS()*, um die Aktualisierungsaktion für jede Instanz durchzuführen, und schläft dann für eine kurze Zeit. Eine andere übliche Aktion in *user_mainloop()* kann der Aufruf der Event-Handler-Schleife eines GUI-Toolkits sein.

- *option userinit yes* - (Voreinstellung: no)
  Diese Option wird ignoriert, wenn die Option *userspace* (siehe oben) auf *no* gesetzt ist. Wenn *userinit* angegeben ist, wird die Funktion *userinit(argc,argv)* vor *rtapi_app_main()* (und damit vor dem Aufruf von *hal_init()*) aufgerufen. Diese Funktion kann die Kommandozeilenargumente verarbeiten oder andere Aktionen ausführen. Ihr Rückgabetyp ist *void*; sie kann *exit()* aufrufen, wenn sie beenden will, anstatt eine HAL-Komponente zu erstellen (z.B. weil die Kommandozeilenargumente ungültig waren).

- *option extra_link_args "..."* - (Voreinstellung: "")
  Diese Option wird ignoriert, wenn die Option *Userspace* (siehe oben) auf *no* gesetzt ist. Beim Linken einer Nicht-Echtzeitkomponente werden die angegebenen Argumente in die Linkzeile eingefügt. Da die Kompilierung in einem temporären Verzeichnis stattfindet, bezieht sich "-L." auf das temporäre Verzeichnis und nicht auf das Verzeichnis, in dem sich die .comp-Quelldatei befindet. Diese Option kann in der halcompile Befehlszeile mit -extra-link-args="-L....." gesetzt werden. Diese Alternative bietet eine Möglichkeit, zusätzliche Flags in Fällen zu setzen, in denen die Eingabedatei eine .c-Datei und keine .comp-Datei ist.
- *option extra_compile_args "..."* - (Voreinstellung: "")
  Diese Option wird ignoriert, wenn die Option *userspace* (siehe oben) auf *no* gesetzt ist. Beim Kompilieren einer Nicht-Echtzeit-Komponente werden die angegebenen Argumente in die Compiler-Befehlszeile eingefügt. Wenn die Eingabedatei eine .c-Datei ist, kann diese Option in der halcompile-Befehlszeile mit --extra-compile-args="-I....." gesetzt werden. Diese Alternative bietet eine Möglichkeit, zusätzliche Flags zu setzen, wenn die Eingabedatei eine .c-Datei und keine .comp-Datei ist.
- *option homemod yes* - (Voreinstellung: no)
  Modul ist ein benutzerdefiniertes Homing-Modul, das mit `[EMCMOT]HOMEMOD=`*Modulname* geladen wird.
- *option tpmod yes* - (Voreinstellung: no)
  Modul ist ein benutzerdefiniertes Trajektorienplanungsmodul (tp), das mit `[TRAJ]TPMOD=`*Modulname* geladen wird.
- *option period no* – (Standard: yes)
  Steuert den impliziten Parameter *period* der in der Komponente definierten Funktion(en). Eine Standardfunktion hat einen impliziten Parameter *period*. Viele Komponenten verwenden den Parameter *period* jedoch nicht, was eine Compiler-Warnung „unused parameter“ verursachen würde. Das Setzen von *option period no* erzeugt eine Funktionsdeklaration ohne den Parameter *period* und verhindert so die Warnung. Das Setzen dieser Option verhindert außerdem die Definition von *fperiod*, da diese von *period* abhängt.

Wenn der VALUE (engl. für Wert) einer Option nicht angegeben wird, ist dies gleichbedeutend mit der Angabe von *option ... yes*.
Das Ergebnis der Zuweisung eines unangemessenen Wertes zu einer Option ist undefiniert
Das Ergebnis der Verwendung einer anderen Option ist undefiniert.

Lizenz und Urheberschaft

- `LICENSE` - Geben Sie die Lizenz des Moduls für die Dokumentation und für die MODULE_LICENSE()-Moduldeklaration an. Zum Beispiel, um anzugeben, dass die Lizenz des Moduls GPL v2 oder höher ist:

  ```
  `license "GPL"; // bedeutet GPL v2 oder höher`
  ```

  Weitere Informationen über die Bedeutung von MODULE_LICENSE() und zusätzliche Lizenzbezeichner finden Sie in *<linux/module.h>* oder in der Handbuchseite zu *rtapi_module_param(3)*.

Diese Erklärung ist **erforderlich**.

- `AUTHOR` - Geben Sie den Autor des Moduls für die Dokumentation an.

Datenspeicherung pro Instanz

- `variable CTYPE STARREDNAME; + variable CTYPE STARREDNAME[SIZE]; + variable CTYPE STARREDNAME = DEFAULT; + variable CTYPE STARREDNAME[SIZE] = DEFAULT;`

  Declare a per-instance variable *STARREDNAME* of type *CTYPE*, optionally as an array of *SIZE* items, and optionally with a default value *DEFAULT*. Items with no *DEFAULT* are initialized to all-bits-zero. *CTYPE* is a simple one-word C type, such as `rtapi_real`, `rtapi_uint`, `rtapi_sint`, `int`, etc. Access to array variables uses square brackets.

Wenn eine Variable ein Zeigertyp sein soll, darf zwischen dem "*" und dem Variablennamen kein Leerzeichen stehen. Daher ist das Folgende akzeptabel:

```
variable int *example;
```

Aber die folgenden sind es nicht:

```
variable int* badexample;
variable int * badexample;
```

Kommentare

Einzeilige Kommentare im C++-Stil (`//...`) und mehrzeilige Kommentare im C-Stil (`/* ... */`) werden beide im Deklarationsabschnitt unterstützt.

5.10.9. Einschränkungen

Obwohl HAL erlaubt, dass ein Pin, ein Parameter und eine Funktion denselben Namen haben können, ist dies bei *halcompile* nicht der Fall.

Zu den Variablen- und Funktionsnamen, die nicht verwendet werden können oder zu Problemen führen können, gehören:

- Alles, was mit *_comp* beginnt.
- *comp_id*
- *fperiod*
- *rtapi_app_main*
- *rtapi_app_exit*
- *extra_setup*
- *extra_cleanup*
- *post_export*

5.10.10. Bequemlichkeits-Makros

Based on the items in the declaration section, *halcompile* creates a C structure called `struct __comp_state`. However, instead of referring to the members of this structure (e.g., `*(inst\->name)`), they will generally be referred to using the macros below. The details of `struct __comp_state` and these macros may change from one version of *halcompile* to the next.

- `FUNCTION(`__name__)` - Verwenden Sie dieses Makro, um die Definition einer Echtzeitfunktion zu beginnen, die zuvor mit *function NAME* deklariert wurde. Die Funktion enthält einen Parameter *period*, der die ganzzahlige Anzahl von Nanosekunden zwischen Aufrufen der Funktion angibt. Siehe auch zuvor beschriebene Option *period*.
- `EXTRA_SETUP()` - Verwenden Sie dieses Makro, um die Definition der Funktion zu beginnen, die aufgerufen wird, um eine zusätzliche Einrichtung dieser Instanz durchzuführen. Geben Sie einen negativen UNIX-*errno*-Wert zurück, um einen Fehler anzuzeigen (z.B. *return -EBUSY*, wenn die Reservierung eines I/O-Ports fehlgeschlagen ist), oder 0, um einen Erfolg anzuzeigen.
- `EXTRA_CLEANUP()` - Verwenden Sie dieses Makro zu Beginn der Definition derjenigen Funktion, die eine Erweiterung des Aufräumen der Komponente implementiert. Beachten Sie, dass diese Funktion alle Instanzen der Komponente aufräumen muss, nicht nur eine. Die Makros "pin_name", "parameter_name" und "data" dürfen hier nicht verwendet werden.
- `POST_EXPORT()` - Use this macro to begin the definition of the function called to perform extra setup *after* all pins and parameters have been created for this instance. This function is called just before the export() function returns. You can use the POST_EXPORT() function to preset parameter values you otherwise would not be able to set. Return a negative UNIX *errno* value to indicate failure, or 0 to indicate success.
  Note: POST_EXPORT() should not change *personality*. If you need to adapt *personality*, then you must do so in EXTRA_SETUP(), which runs before pins and parameters are created.
- *pin_name* oder *parameter_name* - Für jeden Pin *pin_name* oder Parameter *parameter_name* gibt es ein Makro, mit dem der Name allein verwendet werden kann, um auf den Pin oder Parameter zu verweisen. Wenn *pin_name* oder *parameter_name* ein Array ist, hat das Makro die Form *pin_name(idx)* oder *param_name(idx)*, wobei *idx* der Index im Pin-Array ist. Handelt es sich bei dem Array um ein Array mit variabler Größe, ist es nur zulässig, um auf Elemente bis zu seiner *condsize* zu verweisen.

  Wenn es sich um eine bedingte Position handelt, kann nur auf sie verwiesen werden, wenn ihre "Bedingung" einen Wert ungleich Null ergibt.

- *variable_name* - Für jede Variable *variable_name* gibt es ein Makro, das es erlaubt, den Namen allein zu verwenden, um auf die Variable zu verweisen. Wenn *variable_name* ein Array ist, wird das normale C-Subskript verwendet: *variable_name[idx]*.
- *data* - Wenn "option data" angegeben ist, ermöglicht dieses Makro den Zugriff auf die Instanzdaten.
- *fperiod* - Die Gleitkommazahl von Sekunden zwischen Aufrufen dieser Echtzeitfunktion. Siehe auch die zuvor beschriebene Option *period*.
- `FOR_ALL_INSTS() `{...}` - Für Nicht-Echtzeit-Komponenten. Dieses Makro iteriert über alle definierten Instanzen. Innerhalb des Schleifenkörpers arbeiten die Makros *pin_name*, *parameter_name* und *data* wie in Echtzeitfunktionen.

5.10.11. Komponenten mit einer Funktion

Wenn eine Komponente nur eine Funktion hat und die Zeichenkette `"FUNCTION"` nirgendwo nach `;;` auftaucht, dann wird der Teil nach `;;` als der Körper der einzigen Funktion der Komponente angesehen. Siehe Simple Comp für ein Beispiel hierfür.

5.10.12. Komponenten-Persönlichkeit

Wenn eine Komponente Pins oder Parameter mit einer "if-Bedingung" oder "[maxsize : condsize]" hat, wird sie als Komponente mit "Persönlichkeit" bezeichnet. Die "Persönlichkeit" jeder Instanz wird beim Laden des Moduls festgelegt. Die "Persönlichkeit" kann verwendet werden, um Pins nur bei Bedarf zu erstellen. So wird die "Persönlichkeit" beispielsweise in der Komponente *logic* (engl. für Logik) verwendet, um eine variable Anzahl von Eingangspins für jedes Logikgatter und die Auswahl einer der grundlegenden booleschen Logikfunktionen *und*, *oder* und *xor* zu ermöglichen.

Die Standardanzahl der erlaubten "personality"-Elemente ist eine Kompilierzeiteinstellung (64). Die Vorgabe gilt für zahlreiche in der Distribution enthaltene Komponenten, die mit halcompile erstellt werden.

Um die zulässige Anzahl von Persönlichkeitselementen für benutzerdefinierte Komponenten zu ändern, verwenden Sie die Option *--personalities* mit halcompile. Zum Beispiel, um bis zu 128 Persönlichkeitszeiten zu erlauben:

```
  [sudo] halcompile --personalities=128 --install ...
```

Bei der Verwendung von Komponenten mit Persönlichkeit ist es üblich, ein Persönlichkeitselement für **jede** angegebene Komponenteninstanz anzugeben. Beispiel für 3 Instanzen der Logikkomponente:

```
loadrt logic names=and4,or3,nand5, personality=0x104,0x203,0x805
```

NOTE

Wenn eine loadrt-Zeile mehr Instanzen als Persönlichkeiten angibt, wird den Instanzen mit nicht angegebenen Persönlichkeiten eine Persönlichkeit von 0 zugewiesen. Wenn die angeforderte Anzahl von Instanzen die Anzahl der erlaubten Persönlichkeiten übersteigt, werden die Persönlichkeiten durch Indexierung modulo der Anzahl der erlaubten Persönlichkeiten zugewiesen. Es wird eine Meldung über solche Zuweisungen ausgegeben.

NOTE

If a component uses personality, then it should generally check its value. The value of personality is zero if the *personality=N* argument is not provided to loadrt. Pins and params whose personality constraint evaluates to zero are not created, and their memory remains NULL. Unconditionally accessing such a pin or param dereferences a NULL pointer and crashes the realtime process.

You can use a test in EXTRA_SETUP() to test the acceptable values of personality for your component. You should return -EINVAL if your conditions are not met.

Example testing personality:

```
EXTRA_SETUP(){
    // Silence warnings for the unused arguments.
    (void)prefix;
    (void)extra_arg;
    if (personality < 1) {
        rtapi_print_msg(RTAPI_MSG_ERR,
            "mycomp: personality must be >= 1 (use personality=N to set the
number of channels)\n");
        return -EINVAL;
    }
    return 0;
}
```

5.10.13. Beispiele

Konstante

Beachten Sie, dass die Deklaration "function _" Funktionen mit dem Namen "constant.0" usw. erzeugt. Der Dateiname muss mit dem Komponentennamen übereinstimmen.

```
component constant;
pin out real out;
param r real value = 1.0;
option period no;
function _;
license "GPL"; // indicates GPL v2 or later
;;
FUNCTION(_) { out_set(value); }
```

sincos

Diese Komponente berechnet den Sinus und Kosinus eines Eingangswinkels im Bogenmaß. Sie hat andere Fähigkeiten als die "Sinus"- und "Kosinus"-Ausgänge von siggen, weil die Eingabe ein Winkel ist und nicht frei auf der Grundlage eines "Frequenz"-Parameters läuft.

Die Pins werden im Quellcode mit den Namen *sin_* und *cos_* deklariert, damit sie nicht mit den Funktionen *sin()* und *cos()* interferieren. Die HAL-Pins heißen weiterhin *sincos.<num>.sin*.

```
component sincos;
pin out real sin_;
pin out real cos_;
pin in real theta;
option period no;
function _;
license "GPL"; // indicates GPL v2 or later
;;
#include <rtapi_math.h>
FUNCTION(_) {
    sin_=sin(theta));
    cos_=cos(theta));
```

```
}
```

out8

This component is a driver for a *fictional* card called "out8", which has 8 pins of digital output which are treated as a single 8-bit value. There can be a varying number of such cards in the system, and they can be at various addresses. The pin is called *out_* because *out* is an identifier used in *<rtapi_io.h>*. It illustrates the use of *POST_EXPORT* and *EXTRA_CLEANUP* to request an I/O region and then free it in case of error or when the module is unloaded.

```
component out8;
pin out uint out_ "Output value; only low 8 bits are used";
param r uint ioaddr;

function _;

option period no;
option count_function;
option post_export yes;
option extra_cleanup;
option constructable no;

license "GPL"; // indicates GPL v2 or later
;;
#include <rtapi_io.h>

#define MAXOUT 8
int io[MAXOUT] = {};
RTAPI_MP_ARRAY_INT(io, MAXOUT, "I/O addresses of out8 boards");

int get_count(void) {
    int i = 0;
    for(i = 0; i < MAXOUT && io[i]; i++) { /* Nothing */ }
    return i;
}

POST_EXPORT() {
    (void)prefix;
    if(!rtapi_request_region(io[extra_arg], 1, "out8")) {
        // set this I/O port to 0 so that EXTRA_CLEANUP does not release the IO
        // ports that were never requested.
        io[extra_arg] = 0;
        return -EBUSY;
    }
    // This can only be done when the param has been created and that can only
    // be done in POST_EXPORT. Note the EXTRA_SETUP runs *before* the parameter
    // is created and can therefore not set the parameter value.
    ioaddr_set(io[extra_arg]);
    return 0;
}

EXTRA_CLEANUP() {
    for(int i = 0; i < MAXOUT && io[i]; i++) {
        rtapi_release_region(io[i], 1);
```

```
    }
}

FUNCTION(_) { rtapi_outb(out_, ioaddr); }
```

hal_loop

```
component hal_loop;
pin out real example;
```

Dieses Fragment einer Komponente veranschaulicht die Verwendung des Präfixes "hal_" in einem Komponentennamen.

loop ist ein gebräuchlicher Name (in der englischsprachig dominierten Programmierung), und das Präfix *hal_* vermeidet mögliche Namenskollisionen mit anderer, nicht verwandter Software. Zum Beispiel läuft auf RTAI-Echtzeitsystemen Echtzeitcode im Kernel, wenn die Komponente also nur "loop" heißen würde, könnte sie leicht mit dem Standard-Kernelmodul "loop" in Konflikt geraten.

Nach dem Laden zeigt *halcmd show comp* eine Komponente namens *hal_loop* an. Der von "halcmd show pin" angezeigte Pin ist jedoch "loop.0.example" und nicht "hal-loop.0.example".

arraydemo

Diese Echtzeitkomponente veranschaulicht die Verwendung von Arrays fester Größe:

```
component arraydemo "4-bit Shift register";
pin in bool in;
pin out bool out-# [4];
option period no;
function _;
license "GPL"; // indicates GPL v2 or later
;;
for(int i = 3; i > 0; i--) {
    out_set(i, out(i-1));
}
out_set(0, in);
```

rand

Diese Nicht-Echtzeit-Komponente ändert den Wert an ihrem Ausgangspin etwa alle 1 ms auf einen neuen Zufallswert im Bereich (0,1).

```
component rand;
option userspace;

pin out real out;
license "GPL"; // indicates GPL v2 or later
;;
#include <unistd.h>
```

```
void user_mainloop(void) {
    while(1) {
        usleep(1000);
        FOR_ALL_INSTS() out_set(drand48());
    }
}
```

Logik (unter Nutzung einer "Personality")

Diese Echtzeitkomponente zeigt, wie man "Persönlichkeit" verwendet, um Arrays variabler Größe und optionale Pins zu erstellen.

```
component logic "LinuxCNC HAL component providing experimental logic functions";
pin in bool in-##[16 : personality & 0xff];
pin out bool and if personality & 0x100;
pin out bool or if personality & 0x200;
pin out bool xor if personality & 0x400;
option period no;
function _;
description """
Experimental general 'logic function' component.  Can perform 'and', 'or'
and 'xor' of up to 16 inputs.  Determine the proper value for 'personality'
by adding:

* 4 - The number of input pins, usually from 2 to 16
* 256 (0x100) - if the 'and' output is desired
* 512 (0x200) - if the 'or' output is desired
* 1024 (0x400) - if the 'xor' (exclusive or) output is desired

""";
license "GPL"; // indicates GPL v2 or later
;;
FUNCTION(_) {
    rtapi_bool a = 1;
    rtapi_bool o = 0;
    rtapi_bool x = 0;
    for(int i = 0; i < (personality & 0xff); i++) {
        if(in(i)) {
            o = 1;
            x = !x;
        } else {
            a = 0;
        }
    }
    if(personality & 0x100) and_set(a);
    if(personality & 0x200) or_set(o);
    if(personality & 0x400) xor_set(x);
}
```

Eine typische Zeile zur Belegung dieses Bauteil könnte lauten

```
loadrt logic count=3 personality=0x102,0x305,0x503
```

wodurch die folgenden Pins erstellt werden:

- A 2-input AND gate: `logic.0.and`, `logic.0.in-00`, `logic.0.in-01`
- 5-input AND and OR gates: `logic.1.and`, `logic.1.or`, `logic.1.in-00`, `logic.1.in-01`, `logic.1.in-02`, `logic.1.in-03`, `logic.1.in-04`,
- 3-input AND and XOR gates: `logic.2.and`, `logic.2.xor`, `logic.2.in-00`, `logic.2.in-01`, `logic.2.in-02`

Allgemeine Funktionen

Dieses Beispiel zeigt, wie man Funktionen von der Hauptfunktion aus aufruft. Es zeigt auch, wie die Referenz von HAL-Pins an diese Funktionen übergeben werden kann.

```
component example;
pin in sint in;
pin out bool out1;
pin out bool out2;

option period no;
function _;
license "GPL";
;;

// general pin set true function
void doset(hal_bool_t p) {
    hal_set_bool(p, 1);
}

// general pin set false function
void unset(hal_bool_t p) {
    hal_set_bool(p, 0);
}

//main function
FUNCTION(_) {
    rtapi_sint inval = in;
    if (inval < 0) {
        doset(out1_ptr);
        unset(out2_ptr);
    } else if (inval > 0) {
        unset(out1_ptr);
        doset(out2_ptr);
    } else {
        unset(out1_ptr);
        unset(out2_ptr);
    }
}
```

This component uses two general function to manipulate a HAL bool pin referenced to it.

5.10.14. Verwendung der Kommandozeile

Die Manpage zu halcompile enthält Details zum Aufruf von `halcompile`.

```
$ man halcompile
```

Eine kurze Zusammenfassung der Verwendung von halcompile finden Sie hier:

```
$ halcompile --help
```

5.11. HALTCL-Dateien

halcmd zeichnet sich durch die Angabe von Komponenten und Verbindungen aus, aber diese Skripte bieten keine Berechnungsmöglichkeiten. Infolgedessen sind INI-Dateien in der Klarheit und Kürze, die mit höheren Sprachen möglich ist, eingeschränkt.

Die haltcl-Funktionalität bietet die Möglichkeit, Tcl-Skripte und ihre Funktionen für Berechnungen, Schleifen, Verzweigungen, Prozeduren usw. in INI-Dateien zu verwenden. Um diese Funktionalität zu nutzen, verwenden Sie die Tcl-Sprache und die Erweiterung .tcl für HAL-Dateien.

Die Erweiterung .tcl wird von dem Hauptskript (`linuxcnc`) verstanden, das INI-Dateien verarbeitet. Haltcl-Dateien werden im HAL-Abschnitt von INI-Dateien identifiziert (genau wie HAL-Dateien).

Beispiel

```
[HAL]
HALFILE = conventional_file.hal
HALFILE = tcl_based_file.tcl
```

Bei entsprechender Sorgfalt können HAL- und Tcl-Dateien miteinander vermischt werden.

5.11.1. Kompatibilität

Die in HAL-Dateien verwendete halcmd-Sprache hat eine einfache Syntax, die eigentlich eine Teilmenge der leistungsfähigeren Allzweck-Skriptsprache Tcl ist.

5.11.2. Haltcl-Befehle

Haltcl-Dateien verwenden die Tcl-Skriptsprache, die mit den spezifischen Befehlen der LinuxCNC-Hardware-Abstraktionsschicht (HAL) erweitert wird. Die HAL-spezifischen Befehle sind:

```
addf, alias,
delf, delsig,
getp, gets
ptype,
stype,
help,
linkps, linksp, list, loadrt, loadusr, lock,
net, newsig,
```

```
save, setp, sets, show, source, start, status, stop,
unalias, unlinkp, unload, unloadrt, unloadusr, unlock,
waitusr
```

Für die Befehle *gets* und *list* gibt es zwei Sonderfälle aufgrund von Konflikten mit eingebauten Tcl-Befehlen. Für haltcl muss diesen Befehlen das Schlüsselwort *hal* vorangestellt werden:

```
halcmd   haltcl
------   ------
gets     hal gets
list     hal list
```

5.11.3. Haltcl INI-Datei-Variablen

Auf Variablen in INI-Dateien kann sowohl mit `halcmd` als auch mit `haltcl` zugegriffen werden, allerdings mit unterschiedlicher Syntax. LinuxCNC INI-Dateien verwenden SECTION- und ITEM-Spezifikationen, um Konfigurationselemente zu identifizieren:

```
[SECTION_A]
ITEM1 = value_1
ITEM2 = value_2
# ...
[SECTION_B]
# ...
```

Die Werte der INI-Datei sind durch Textersetzung in HAL-Dateien in folgender Form zugänglich:

```
[SECTION]ITEM
```

Die gleichen Werte der INI-Datei sind in Tcl-Dateien in Form einer globalen Tcl-Array-Variable zugänglich:

```
$::SECTION(ITEM)
```

Zum Beispiel, ein INI-Datei Element wie:

```
[JOINT_0]
MAX_VELOCITY = 4
```

wird in HAL-Dateien für halcmd als *[JOINT_0]MAX_VELOCITY* ausgedrückt
und als *$::JOINT_0(MAX_VELOCITY)* in Tcl-Dateien für haltcl.

Da INI-Dateien das gleiche ITEM in der gleichen SECTION mehrfach wiederholen kann, ist $::SECTION(ITEM) eigentlich eine Tcl-Liste jedes einzelnen Wertes.

Wenn es nur einen Wert gibt und dieser ein einfacher Wert ist (alle Werte, die nur aus Buchstaben und Zahlen ohne Leerzeichen bestehen, gehören zu dieser Gruppe), dann ist es möglich, `$::SECTION(ITEM)` so zu behandeln, als ob es keine Liste wäre.

Wenn der Wert Sonderzeichen enthalten könnte (Anführungszeichen, geschweifte Klammern, eingebettete Leerzeichen und andere Zeichen, die in Tcl eine besondere Bedeutung haben), dann ist es notwendig, zwischen der Liste der Werte und dem ersten (und möglicherweise einzigen) Wert in der Liste zu unterscheiden.

In Tcl wird dies als `[lindex $::SECTION(ITEM) 0]` geschrieben.

Beispiel: Bei den folgenden INI-Werten

```
[HOSTMOT2]
DRIVER=hm2_eth
IPADDR="10.10.10.10"
BOARD=7i92
CONFIG="num_encoders=0 num_pwmgens=0 num_stepgens=6"
```

Und diesem loadrt-Befehl:

```
loadrt $::HOSTMOT2(DRIVER) board_ip=$::HOSTMOT2(IPADDR) config=$::HOSTMOT2(CONFIG)
```

Ist dieses der eigentliche Befehl, der ausgeführt wird:

```
loadrt hm2_eth board_ip={"10.10.10.10"} config={"num_encoders=0 num_pwmgens=0
num_stepgens=6"}
```

Dies schlägt fehl, weil loadrt die geschweiften Klammern nicht erkennt.

Um die Werte so zu erhalten, wie sie in der INI-Datei eingegeben wurden, schreiben Sie die loadrt-Zeile wie folgt um:

```
loadrt $::HOSTMOT2(DRIVER) board_ip=[lindex $::HOSTMOT2(IPADDR) 0] config=[lindex
$::HOSTMOT2(CONFIG) 0]
```

5.11.4. Konvertieren von HAL-Dateien in Tcl-Dateien

Vorhandene HAL-Dateien können durch manuelle Bearbeitung in Tcl-Dateien konvertiert werden, um die oben genannten Unterschiede zu berücksichtigen. Der Prozess kann mit Skripten automatisiert werden, die diese Substitutionen verwenden.

```
[SECTION]ITEM ---> $::SECTION(ITEM)
gets          ---> hal gets
list          ---> hal list
```

5.11.5. Haltcl Anmerkungen

In haltcl wird das Argument value für die Befehle *sets* und *setp* implizit als Ausdruck in der Tcl-Sprache behandelt.

Beispiel

```
# Verstärkung für die Umrechnung von Grad/Sekunde in Einheiten/Minute für den JOINT_0-
Radius festlegen
setp scale.0.gain 6.28/360.0*$::JOINT_0(radius)*60.0
```

Leerzeichen im bloßen Ausdruck sind nicht erlaubt, verwenden Sie dafür Anführungszeichen:

```
setp scale.0.gain "6.28 / 360.0 * $::JOINT_0(radius) * 60.0"
```

In anderen Zusammenhängen, wie z. B. bei *loadrt*, müssen Sie den Tcl "expr"-Befehl ([expr {}]) ausdrücklich für Berechnungsausdrücke verwenden.

Beispiel

```
loadrt motion base_period=[expr {500000000/$::TRAJ(MAX_PULSE_RATE)}]
```

5.11.6. Haltcl Beispiele

Betrachten Sie das Thema "Stepgen Headroom". Die Software `stepgen` läuft am besten mit einer Beschleunigungsbeschränkung, die "ein bisschen höher" ist als die vom Bewegungsplaner verwendete. Bei der Verwendung von `halcmd`-Dateien erzwingen wir daher, dass INI-Dateien einen manuell berechneten Wert haben.

```
[JOINT_0]
MAXACCEL = 10.0
STEPGEN_MAXACCEL = 10.5
```

Mit `haltcl` können Sie Tcl-Befehle verwenden, um die Berechnungen durchzuführen, und das `STEPGEN_MAXACCEL`-Element in der INI-Datei ganz eliminieren:

```
setp stepgen.0.maxaccel $::JOINT_0(MAXACCEL)*1.05
```

Eine weitere Funktion von `haltcl` ist die Schleifenbildung und das Testen. Viele Simulatorkonfigurationen verwenden zum Beispiel die HAL-Dateien "core_sim.hal" oder "core_sim9.hal". Diese unterscheiden sich durch die Anforderung, mehr oder weniger Achsen anzuschließen. Der folgende haltcl-Code würde für jede Kombination von Achsen in einer trivkins-Maschine funktionieren.

```
# Anlegen von position, velocity (Geschwindigkeit) and acceleration (Beschleunigung)
Signalen für jede Achse
set ddt 0
for {set jnum 0} {$jnum < $::KINS(JOINTS)} {incr jnum} {
  # 'list pin' gibt eine leere Liste zurück, wenn der Pin nicht existiert
  if {[hal list pin joint.${jnum}.motor-pos-cmd] == {}} {
    continue
  }
  net ${jnum}pos joint.${jnum}.motor-pos-cmd => joint.$axno.motor-pos-fb \
                                             => ddt.$ddt.in
  net ${axis}vel <= ddt.$ddt.out
```

```
  incr ddt
  net ${axis}vel => ddt.$ddt.in
  net ${axis}acc <= ddt.$ddt.out
  incr ddt
}
puts [show sig *vel]
puts [show sig *acc]
```

5.11.7. Haltcl Interaktiv

Der Befehl `halrun` erkennt haltcl-Dateien. Mit der Option -T kann haltcl interaktiv als Tcl-Interpreter ausgeführt werden. Diese Fähigkeit ist nützlich zum Testen und für eigenständige HAL-Anwendungen.

Beispiel

```
$ halrun -T haltclfile.tcl
```

5.11.8. Haltcl-Verteilungsbeispiele (sim)

Das Verzeichnis configs/sim/axis/simtcl enthält eine INI-Datei, die eine .tcl-Datei verwendet, um eine haltcl-Konfiguration in Verbindung mit der Verwendung der twopass-Verarbeitung zu demonstrieren. Das Beispiel zeigt die Verwendung von Tcl-Prozeduren, Schleifen, die Verwendung von Kommentaren und die Ausgabe auf dem Terminal.

5.12. HAL-Benutzeroberfläche

5.12.1. Einführung

Halui ist eine HAL-basierte Benutzeroberfläche für LinuxCNC, es verbindet HAL-Pins mit NML-Befehlen. Die meisten Funktionen (Schaltflächen, Anzeigen usw.), wie von einer traditionellen GUI (AXIS, GMOCCAPY, QtDragon, etc.) zur Verfügung gestellt, werden von HAL-Pins in Halui übernommen.

Der einfachste Weg, halui hinzuzufügen, besteht darin, das Folgende in den [HAL]-Abschnitt der INI-Datei einzufügen:

```
[HAL]
HALUI = halui
```

Ein alternativer Weg, es aufzurufen (besonders, wenn Sie die Konfiguration mit StepConf erzeugen), ist, das Folgende in Ihre `custom.hal`-Datei aufzunehmen.
Stellen Sie sicher, dass Sie den richtigen Pfad zu Ihrer INI-Datei verwenden.

```
loadusr halui -ini /path/to/inifile.ini
```

5.12.2. MDI

Manchmal möchte der Benutzer kompliziertere Aufgaben hinzufügen, die durch die Aktivierung eines

HAL-Pins ausgeführt werden sollen. Dies ist durch Hinzufügen von MDI-Befehlen in die INI-Datei im Abschnitt [HALUI] möglich. Beispiel:

```
[HALUI]
MDI_COMMAND = G0 X0
MDI_COMMAND = G0 G53 Z0
MDI_COMMAND = G28
MDI_COMMAND = o<mysub>call
# ...
```

Wenn halui startet, liest es die 'MDI_COMMAND'-Felder in der INI und exportiert Pins von 00 bis zur Anzahl der `MDI_COMMAND's, die in der INI gefunden wurden, bis zu einem Maximum von 64 Befehlen. Diese Pins können wie alle HAL-Pins angeschlossen werden. Eine gängige Methode ist die Verwendung von Schaltflächen, die von virtuellen Bedienfeldern bereitgestellt werden, wie im Beispiel für MDI_COMMAND Verbindungen gezeigt.

Example 3. Beispiel für MDI_COMMAND-Verbindungen

HAL-Datei

```
net quill-up      halui.mdi-command-00 <= pyvcp.quillup
net reference-pos halui.mdi-command-01 <= pyvcp.referencepos
net call-mysub    halui.mdi-command-02 <= pyvcp.callmysub
```

Netze zum Verbinden der von halui bereitgestellten `halui.mdi-command-NN`*-Pins.*

```
$ halcmd show pin halui.mdi
Component Pins:
Owner  Type  Dir Value  Name
   10  bool  IN  FALSE  halui.mdi-command-00 <== quill-up
   10  bool  IN  FALSE  halui.mdi-command-01 <== reference-pos
   10  bool  IN  FALSE  halui.mdi-command-02 <== call-mysub
    ...
```

Wenn ein Halui-MDI-Pin auf true gesetzt (gepulst) wird, sendet halui den in der INI definierten MDI-Befehl. Dies wird je nach aktuellem Betriebsmodus nicht immer gelingen (z.B. während in AUTO halui MDI-Befehle nicht erfolgreich senden kann).

5.12.3. Beispiel-Konfiguration

Eine Beispiel-Sim-Konfiguration (`configs/sim/axis/halui_pyvcp/halui.ini`) ist in der Distribution enthalten.

5.12.4. Halui-Pin-Referenz

Alle halui-Pins sind auch in der halui-Manualseite dokumentiert:

```
$ man halui
```

Oder siehe http://linuxcnc.org/docs/devel/html/man/man1/halui.1.html

Abbrechen

- *halui.abort* (bool, in) - pin to send an abort message (clears out most errors)

Notaus (engl. E-Stop)

- *halui.estop.activate* (bool, in) - pin for requesting E-Stop
- *halui.estop.is-activated* (bool, out) - indicates E-stop reset
- *halui.estop.reset* (bool, in) - pin for requesting E-Stop reset

Vorschub Neufestsetzung (engl. override)

- *halui.feed-override.count-enable* (bool, in) - must be true for *counts* or *direct-value* to work.
- *halui.feed-override.counts* (sint, in) - counts * scale = FO percentage. Can be used with an encoder or *direct-value*.
- *halui.feed-override.decrease* (bool, in) - pin for decreasing the FO (-=scale)
- *halui.feed-override.increase* (bool, in) - pin for increasing the FO (+=scale)
- *halui.feed-override.reset* (bool, in) - pin for resetting the FO (scale=1.0)
- *halui.feed-override.direct-value* (bool, in) - false when using encoder to change counts, true when setting counts directly.
- *halui.feed-override.scale* (real, in) - pin for setting the scale for increase and decrease of *feed-override*.
- *halui.feed-override.value* (real, out) - current FO value

Nebel (engl. mist)

- *halui.mist.is-on* (bool, out) - indicates mist is on
- *halui.mist.off* (bool, in) - pin for requesting mist off
- *halui.mist.on* (bool, in) - pin for requesting mist on

Flut-Kühlmittel (engl. flood)

- *halui.flood.is-on* (bool, out) - indicates flood is on
- *halui.flood.off* (bool, in) - pin for requesting flood off
- *halui.flood.on* (bool, in) - pin for requesting flood on

Referenzfahrt (engl. homing)

- *halui.home-all* (bool, in) - pin for requesting all axis to home. This pin will only be there if HOME_SEQUENCE is set in the INI file.

Maschine

- *halui.machine.units-per-mm* (real out) - pin for machine units-per-mm (inch:1/25.4, mm:1) according to inifile setting: [TRAJ]LINEAR_UNITS
- *halui.machine.is-on* (bool, out) - indicates machine on
- *halui.machine.off* (bool, in) - pin for requesting machine off
- *halui.machine.on* (bool, in) - pin for requesting machine on

Max. Geschwindigkeit

Die maximale lineare Geschwindigkeit kann zwischen 0 und der MAX_VELOCITY eingestellt werden, die im Abschnitt [TRAJ] der INI-Datei festgelegt ist.

- *halui.max-velocity.count-enable* (bool, in) - must be true for *counts* or *direct-value* to work.
- *halui.max-velocity.counts* (sint, in) - counts * scale = MV percentage. Can be used with an encoder or *direct-value*.
- *halui.max-velocity.direct-value* (bool, in) - false when using encoder to change counts, true when setting counts directly.
- *halui.max-velocity.decrease* (bool, in) - pin for decreasing max velocity
- *halui.max-velocity.increase* (bool, in) - pin for increasing max velocity
- *halui.max-velocity.scale* (real, in) - the amount applied to the current maximum velocity with each transition from off to on of the increase or decrease pin in machine units per second.
- *halui.max-velocity.value* (real, out) - is the maximum linear velocity in machine units per second.

MDI

- *halui.mdi-command-<nn>* (bool, in) - halui will try to send the MDI command defined in the INI. <*nn*> is a two digit number starting at 00.
  If the command succeeds then it will place LinuxCNC in the MDI mode and then back to Manual mode.
  If no [HALUI]MDI_COMMAND variables are set in the ini file, no halui.mdi-command-<nn> pins will be exported by halui.
- *halui.halui-mdi-is-running* (bool, out) - execution status of MDI commands sent by halui. The status is active even during mode switching. If no [HALUI]MDI_COMMAND variables are set in the ini file, this pins will not be exported by halui.

Gelenk

N = Gelenknummer (0 ... num_joints-1)
Beispiel:

- *halui.joint.N.select* (bool in) - pin for selecting joint *N*
- *halui.joint.N.is-selected* (bool out) - status pin that joint *N* is selected

- *halui.joint.N.has-fault* (bool out) - status pin telling that joint *N* has a fault
- *halui.joint.N.home* (bool in) - pin for homing joint *N*
- *halui.joint.N.is-homed* (bool out) - status pin telling that joint *N* is homed
- *halui.joint.N.on-hard-max-limit* (bool out) - status pin telling that joint *N* is on the positive hardware limit
- *halui.joint.N.on-hard-min-limit* (bool out) - status pin telling that joint *N* is on the negative hardware limit
- *halui.joint.N.on-soft-max-limit* (bool out) - status pin telling that joint *N* is on the positive software limit
- *halui.joint.N.on-soft-min-limit* (bool out) - status pin telling that joint *N* is on the negative software limit
- *halui.joint.N.override-limits* (bool out) - status pin telling that joint *N*'s limits are temporarily overridden
- *halui.joint.N.unhome* (bool in) - pin for unhoming joint *N*
- *halui.joint.selected* (uint out) - selected joint number (0 ... num_joints-1)
- *halui.joint.selected.has-fault* (bool out) - status pin selected joint is faulted
- *halui.joint.selected.home* (bool in) - pin for homing the selected joint
- *halui.joint.selected.is-homed* (bool out) - status pin telling that the selected joint is homed
- *halui.joint.selected.on-hard-max-limit* (bool out) - status pin telling that the selected joint is on the positive hardware limit
- *halui.joint.selected.on-hard-min-limit* (bool out) - status pin telling that the selected joint is on the negative hardware limit
- *halui.joint.selected.on-soft-max-limit* (bool out) - status pin telling that the selected joint is on the positive software limit
- *halui.joint.selected.on-soft-min-limit* (bool out) - status pin telling that the selected joint is on the negative software limit
- *halui.joint.selected.override-limits* (bool out) - status pin telling that the selected joint's limits are temporarily overridden
- *halui.joint.selected.unhome* (bool in) - pin for unhoming the selected joint

Gelenk-Joggen

N = Anzahl der Gelenke (0 ... num_joints-1)

- *halui.joint.jog-deadband* (real in) - pin for setting jog analog deadband (jog analog inputs smaller/slower than this - in absolute value - are ignored)
- *halui.joint.jog-speed* (real in) - pin for setting jog speed for plus/minus jogging.
- *halui.joint.N.analog* (real in) - pin for jogging the joint *N* using a real value (e.g. joy-stick). The value, typically set between 0.0 and ±1.0, is used as a jog-speed multiplier.

- *halui.joint.N.increment* (real in) - pin for setting the jog increment for joint *N* when using increment-plus/minus
- *halui.joint.N.increment-minus* (bool in) - a rising edge will will make joint *N* jog in the negative direction by the increment amount
- *halui.joint.N.increment-plus* (bool in) - a rising edge will will make joint *N* jog in the positive direction by the increment amount
- *halui.joint.N.minus* (bool in) - pin for jogging joint *N* in negative direction at the halui.joint.jog-speed velocity
- *halui.joint.N.plus* (bool in) - pin for jogging joint *N* in positive direction at the halui.joint.jog-speed velocity
- *halui.joint.selected.increment* (real in) - pin for setting the jog increment for the selected joint when using increment-plus/minus
- *halui.joint.selected.increment-minus* (bool in) - a rising edge will will make the selected joint jog in the negative direction by the increment amount
- *halui.joint.selected.increment-plus* (bool in) - a rising edge will will make the selected joint jog in the positive direction by the increment amount
- *halui.joint.selected.minus* (bool in) - pin for jogging the selected joint in negative direction at the halui.joint.jog-speed velocity
- *halui.joint.selected.plus* (bool in) - pin for jogging the selected joint in positive direction at the halui.joint.jog-speed velocity

Achse

L = Buchstabe der Achse (xyzabcuvw)

- *halui.axis.L.select* (bool in) - pin for selecting axis by letter
- *halui.axis.L.is-selected* (bool out) - status pin that axis *L* is selected
- *halui.axis.L.pos-commanded* (real out) - Commanded axis position in machine coordinates
- *halui.axis.L.pos-feedback* (real out) - Feedback axis position in machine coordinates
- *halui.axis.L.pos-relative* (real out) - Feedback axis position in relative coordinates

Achsen-Jogging

L = Buchstabe der Achse (xyzabcuvw)

- *halui.axis.jog-deadband* (real in) - pin for setting jog analog deadband (jog analog inputs smaller/slower than this (in absolute value) are ignored)
- *halui.axis.jog-speed* (real in) - pin for setting jog speed for plus/minus jogging.
- *halui.axis.L.analog* (real in) - pin for jogging the axis *L* using a real value (e.g. joystick). The value, typically set between 0.0 and ±1.0, is used as a jog-speed multiplier.
- *halui.axis.L.increment* (real in) - pin for setting the jog increment for axis *L* when using increment-

plus/minus

- *halui.axis.L.increment-minus* (bool in) - a rising edge will will make axis *L* jog in the negative direction by the increment amount
- *halui.axis.L.increment-plus* (bool in) - a rising edge will will make axis *L* jog in the positive direction by the increment amount
- *halui.axis.L.minus* (bool in) - pin for jogging axis *L* in negative direction at the halui.axis.jog-speed velocity
- *halui.axis.L.plus* (bool in) - pin for jogging axis *L* in positive direction at the halui.axis.jog-speed velocity
- *halui.axis.selected* (uint out) - selected axis (by index: 0:x 1:y 2:z 3:a 4:b 5:cr 6:u 7:v 8:w)
- *halui.axis.selected.increment* (real in) - pin for setting the jog increment for the selected axis when using increment-plus/minus
- *halui.axis.selected.increment-minus* (bool in) - a rising edge will will make the selected axis jog in the negative direction by the increment amount
- *halui.axis.selected.increment-plus* (bool in) - a rising edge will will make the selected axis jog in the positive direction by the increment amount
- *halui.axis.selected.minus* (bool in) - pin for jogging the selected axis in negative direction at the halui.axis.jog-speed velocity
- *halui.axis.selected.plus* (bool in) - for jogging the selected axis in positive direction at the halui.axis.jog-speed velocity

Modus

- *halui.mode.auto* (bool, in) - pin for requesting auto mode
- *halui.mode.is-auto* (bool, out) - indicates auto mode is on
- *halui.mode.is-joint* (bool, out) - indicates joint by joint jog mode is on
- *halui.mode.is-manual* (bool, out) - indicates manual mode is on
- *halui.mode.is-mdi* (bool, out) - indicates MDI mode is on
- *halui.mode.is-teleop* (bool, out) - indicates coordinated jog mode is on
- *halui.mode.joint* (bool, in) - pin for requesting joint by joint jog mode
- *halui.mode.manual* (bool, in) - pin for requesting manual mode
- *halui.mode.mdi* (bool, in) - pin for requesting MDI mode
- *halui.mode.teleop* (bool, in) - pin for requesting coordinated jog mode

Programm

- *halui.program.block-delete.is-on* (bool, out) - status pin telling that block delete is on
- *halui.program.block-delete.off* (bool, in) - pin for requesting that block delete is off
- *halui.program.block-delete.on* (bool, in) - pin for requesting that block delete is on

- *halui.program.is-idle* (bool, out) - status pin telling that no program is running
- *halui.program.is-paused* (bool, out) - status pin telling that a program is paused
- *halui.program.is-running* (bool, out) - status pin telling that a program is running
- *halui.program.optional-stop.is-on* (bool, out) - status pin telling that the optional stop is on
- *halui.program.optional-stop.off* (bool, in) - pin requesting that the optional stop is off
- *halui.program.optional-stop.on* (bool, in) - pin requesting that the optional stop is on
- *halui.program.pause* (bool, in) - pin for pausing a program
- *halui.program.resume* (bool, in) - pin for resuming a paused program
- *halui.program.run* (bool, in) - pin for running a program
- *halui.program.step* (bool, in) - pin for stepping in a program
- *halui.program.stop* (bool, in) - pin for stopping a program

Eilgang-Override (engl. rapid override)

- *halui.rapid-override.count-enable* (bool in (default: TRUE)) - When TRUE, modify Rapid Override when counts changes.
- *halui.rapid-override.counts* (sint in) - counts X scale = Rapid Override percentage. Can be used with an encoder or *direct-value*.
- *halui.rapid-override.decrease* (bool in) - pin for decreasing the Rapid Override (-=scale)
- *halui.rapid-override.direct-value* (bool in) - pin to enable direct value Rapid Override input
- *halui.rapid-override.increase* (bool in) - pin for increasing the Rapid Override (+=scale)
- *halui.rapid-override.scale* (real in) - pin for setting the scale on changing the Rapid Override
- *halui.rapid-override.value* (real out) - current Rapid Override value
- *halui.rapid-override.reset* (bool, in) - pin for resetting the Rapid Override value (scale=1.0)

Spindel Neufestsetzung (engl. override)

- *halui.spindle.N.override.count-enable* (bool, in) - must be true for *counts* or *direct-value* to work.
- *halui.spindle.N.override.counts* (sint, in) - counts * scale = SO percentage. Can be used with an encoder or *direct-value*.
- *halui.spindle.N.override.decrease* (bool, in) - pin for decreasing the SO (-=scale)
- *halui.spindle.N.override.direct-value* (bool, in) - false when using encoder to change counts, true when setting counts directly.
- *halui.spindle.N.override.increase* (bool, in) - pin for increasing the SO (+=scale)
- *halui.spindle.N.override.scale* (real, in) - pin for setting the scale on changing the SO
- *halui.spindle.N.override.value* (real, out) - current SO value
- *halui.spindle.N.override.reset* (bool, in) - pin for resetting the SO value (scale=1.0)

Spindel

- *halui.spindle.N.brake-is-on* (bool, out) - indicates brake is on
- *halui.spindle.N.brake-off* (bool, in) - pin for deactivating spindle/brake
- *halui.spindle.N.brake-on* (bool, in) - pin for activating spindle-brake
- *halui.spindle.N.decrease* (bool, in) - decreases spindle speed
- *halui.spindle.N.forward* (bool, in) - starts the spindle with CW motion
- *halui.spindle.N.increase* (bool, in)- increases spindle speed
- *halui.spindle.N.is-on* (bool, out) - indicates spindle is on (either direction)
- *halui.spindle.N.reverse* (bool, in)- starts the spindle with a CCW motion
- *halui.spindle.N.runs-backward* (bool, out) - indicates spindle is on, and in reverse
- *halui.spindle.N.runs-forward* (bool, out) - indicates spindle is on, and in forward
- *halui.spindle.N.start* (bool, in) - starts the spindle
- *halui.spindle.N.stop* (bool, in) - stops the spindle

Werkzeug

- *halui.tool.length-offset.a* (real out) - current applied tool length offset for the A axis
- *halui.tool.length-offset.b* (real out) - current applied tool length offset for the B axis
- *halui.tool.length-offset.c* (real out) - current applied tool length offset for the C axis
- *halui.tool.length-offset.u* (real out) - current applied tool length offset for the U axis
- *halui.tool.length-offset.v* (real out) - current applied tool length offset for the V axis
- *halui.tool.length-offset.w* (real out) - current applied tool length offset for the W axis
- *halui.tool.length-offset.x* (real out) - current applied tool length offset for the X axis
- *halui.tool.length-offset.y* (real out) - current applied tool length offset for the Y axis
- *halui.tool.length-offset.z* (real out) - current applied tool length offset for the Z axis
- *halui.tool.diameter* (real out) - Current tool diameter, or 0 if no tool is loaded.
- *halui.tool.number* (uint, out) - indicates current selected tool

5.13. Halui Beispiele

Damit alle Halui-Beispiele funktionieren, müssen Sie die folgende Zeile in den [HAL]-Abschnitt der INI-Datei einfügen.

```
HALUI = halui
```

5.13.1. Ferngesteuerter Start

Um einen Fernstartknopf mit LinuxCNC zu verbinden, benutzen Sie den `halui.program.run` Pin und den `halui.mode.auto` Pin. Sie müssen sicherstellen, dass es OK ist, zuerst zu laufen, indem Sie die `halui.mode.is-auto` Pin. Dies geschieht mit einer `and2` Komponente. Die folgende Abbildung zeigt, wie das gemacht wird. Wenn der Fernbedienungsknopf gedrückt wird, ist er sowohl mit `halui.mode.auto` als auch mit `and2.0.in0` verbunden. Wenn der Automodus OK ist, wird der Pin `halui.mode.is-auto` eingeschaltet. Wenn beide Eingänge an der Komponente `and2.0` eingeschaltet sind, wird `and2.0.out` eingeschaltet und das Programm gestartet.

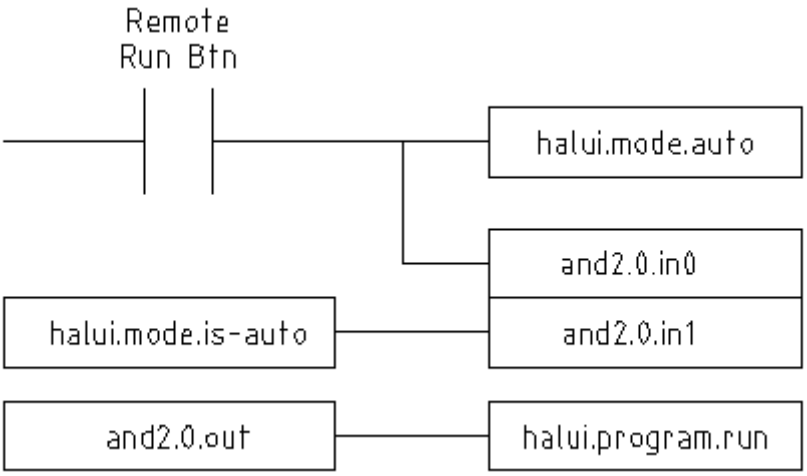


Figure 100. Beispiel für Fernstart

Die für das Vorstehende erforderlichen HAL-Befehle sind:

```
net program-start-btn halui.mode.auto and2.0.in0 <= <your input pin>
net program-run-ok and2.0.in1 <= halui.mode.is-auto
net remote-program-run halui.program.run <= and2.0.out
```

Beachten Sie, dass es in der ersten Zeile zwei Leser-Pins gibt, die auch in zwei Zeilen aufgeteilt werden können:

```
net program-start-btn halui.mode.auto <= <your input pin>
net program-start-btn and2.0.in0
```

5.13.2. Pause & Fortsetzen

Dieses Beispiel wurde entwickelt, um LinuxCNC zu ermöglichen, eine Drehachse auf ein Signal von einer externen Maschine zu bewegen. Die Koordination zwischen den beiden Systemen wird durch zwei Halui Komponenten bereitgestellt:

- `halui.program.is-paused`
- `halui.program.resume`

In Ihrem benutzerdefinierten HAL Datei, fügen Sie die folgenden zwei Zeilen, die mit Ihrer E/A verbunden werden, um auf das Programm Pause oder fortzusetzen, wenn das externe System LinuxCNC fortzusetzen will.

```
net ispaused halui.program.is paused => "Dein output (Ausgabe) Pin"
net resume halui.program.resume <= "your input (Eingabe) Pin"
```

Ihre Eingangs- und Ausgangspins sind mit den Pins verbunden, die mit dem anderen Controller verdrahtet sind. Dabei kann es sich um Pins des Parallelports oder andere E/A-Pins handeln, auf die Sie Zugriff haben.

Dieses System funktioniert auf folgende Weise. Wrid ein M0 in Ihrem G-Code erreicht, so wird das Signal "halui.program.is-paused" wahr. Dies schaltet auf Ihrem Ausgangspin, so dass die externe Steuerung weiß, dass LinuxCNC pausiert ist.

Um die LinuxCNC G-Code-Programm fortzusetzen, wenn die externe Steuerung bereit ist, wird es seine Ausgabe wahr zu machen. Dies wird LinuxCNC signalisieren, dass es die Ausführung von G-Code fortsetzen sollte.

Schwierigkeiten beim Timing

- Das "Resume"-Eingangsrücksignal sollte nicht länger sein als die Zeit, die benötigt wird, um den G-Code wieder zum Laufen zu bringen.
- Der Ausgang "is-paused" sollte nicht mehr aktiv sein, wenn das "resume"-Signal endet.

Diese Timing-Probleme könnten vermieden werden, indem ClassicLadder verwendet wird, um den "is-paused"-Ausgang über einen monostabilen Timer zu aktivieren und einen schmalen Ausgangsimpuls zu liefern. Der "Resume"-Puls könnte ebenfalls über einen monostabilen Timer empfangen werden.

5.14. Anlegen von Nicht-Echtzeit Python Komponenten

In diesem Abschnitt werden die Grundsätze für die Implementierung von HAL-Komponenten mit der Programmiersprache Python erläutert.

5.14.1. Grundlegendes Anwendungsbeispiel

Eine Nicht-Echtzeit-Komponente beginnt mit der Erstellung ihrer Pins und Parameter und tritt dann in eine Schleife ein, die periodisch alle Ausgänge von den Eingängen steuert. Die folgende Komponente kopiert den Wert an ihrem Eingangspin (*passthrough.in*) etwa einmal pro Sekunde an ihren Ausgangspin (*passthrough.out*).

```
#!/usr/bin/env python3
import hal
import time
h = hal.component("passthrough")
h.newpin("in", hal.Type.REAL, hal.Dir.IN)
h.newpin("out", hal.Type.REAL, hal.Dir.OUT)
h.ready()
try:
    while True:
        h['out'] = h['in']
        time.sleep(1)
except KeyboardInterrupt:
```

```
    raise SystemExit
```

Copy the above listing into a file named "passthrough", make it executable (*chmod +x*). Then try it out (assuming the cpmponent "passthrough" is located in the current directory):

Bildschirmkopie mit Einzelheiten über die Ausführung des neu erstellten Passthrough-HAL-Moduls.

```
$ halrun

halcmd: loadusr ./passthrough

halcmd: show pin

Component Pins:
Owner   Type  Dir     Value  Name
     4  real  IN          0  passthrough.in
     4  real  OUT         0  passthrough.out

halcmd: setp passthrough.in 3.14

halcmd: show pin

Component Pins:
Owner   Type  Dir     Value  Name
     4  real  IN       3.14  passthrough.in
     4  real  OUT      3.14  passthrough.out
```

5.14.2. Nicht-Echtzeit-Komponenten und Verzögerungen

Wenn Sie schnell "show pin" eintippen, sehen Sie vielleicht, dass "passthrough.out" immer noch den alten Wert von 0 hat. Das liegt an dem Aufruf von "time.sleep(1)", der dafür sorgt, dass die Zuweisung an den Ausgangspin höchstens einmal pro Sekunde erfolgt. Da es sich um eine Nicht-Echtzeit-Komponente handelt, kann die tatsächliche Verzögerung zwischen den Zuweisungen viel länger sein, wenn der von der Passthrough-Komponente verwendete Speicher auf die Festplatte ausgelagert wird, da die Zuweisung verzögert werden könnte, bis dieser Speicher wieder ausgelagert wird.

So eignen sich Nicht-Echtzeit-Komponenten für benutzerinteraktive Elemente wie Bedienfelder (Verzögerungen im Bereich von Millisekunden werden nicht bemerkt, und längere Verzögerungen sind akzeptabel), nicht aber für das Senden von Schrittimpulsen an eine Stepper-Treiberkarte (Verzögerungen müssen immer im Bereich von Mikrosekunden liegen, egal wie).

5.14.3. Pins und Parameter erstellen

Create a HAL component

```
mycomp = hal.component("passthrough")
```

Die Komponente selbst wird durch einen Aufruf des Konstruktors *hal.component* erzeugt. Die Argumente sind der HAL-Komponentenname und (optional) das für Pin- und Parameternamen verwendete Präfix. Wird das Präfix nicht angegeben, wird der Komponentenname verwendet.

Create a HAL pin or parameter

```
mycomp.newpin("in", hal.Type.REAL, hal.Dir.IN)
mycomp.newparam("has-feature-rotate", hal.Type.BOOL, hal.Dir.RO)
```

Dann werden Pins durch Aufrufe von Methoden auf dem Komponentenobjekt erstellt. Die Argumente sind: Suffix des Pin-Namens, Pin-Typ und Pin-Richtung. Bei Parametern lauten die Argumente: Parameternamenssuffix, Parametertyp und Parameterrichtung.

Table 13. Namen der HAL-Optionen

Pin- und Parametertypen:	`hal.Type.BOOL`	`hal.Type.REAL`	`hal.Type.SINT`	`hal.Type.UINT`	`hal.Type.PORT`
Pin-Richtungen:	`hal.Dir.IN`	`hal.Dir.OUT`	`hal.Dir.IO`		
Parameter-Richtungen:	`hal.Dir.RO`	`hal.Dir.RW`			

NOTE Only pins and signals can use the `hal.Type.PORT` type.

Der vollständige Pin- oder Parametername wird durch Verbinden des Präfixes und des Suffixes mit einem "." gebildet. Im Beispiel heißt der erstellte Pin also *passthrough.in*.

```
mycomp.ready()
```

Sobald alle Pins und Parameter erstellt wurden, rufen Sie die Methode *.ready()* auf.

Ändern des Präfixes

Das Präfix kann durch den Aufruf der Methode *.setprefix()* geändert werden. Das aktuelle Präfix kann durch den Aufruf der Methode *.getprefix()* abgefragt werden.

5.14.4. Lesen und Schreiben von Pins und Parametern

Bei Pins und Parametern, die auch echte Python-Bezeichner sind, kann der Wert unter Verwendung der Attributsyntax aufgerufen oder gesetzt werden:

```
mycomp.out = mycomp.in
```

Für alle Pins, unabhängig davon, ob sie auch echte Python-Bezeichner sind oder nicht, kann der Wert unter Verwendung der tiefgestellten Syntax aufgerufen oder gesetzt werden:

```
mycomp['out'] = mycomp['in']
```

Um alle Pins mit ihren Werten zu sehen, gibt getpins alle Werte in einem Wörterbuch dieser Komponente zurück.

```
mycomp.getpins()
>>>{'in': 0.0, 'out': 0.0}
```

Ansteuerung der Ausgangsstifte (HAL_OUT)

Periodically, usually in response to a timer, all `hal.Dir.OUT` pins should be "driven" by assigning them a new value. This should be done whether or not the value is different than the last one assigned. When a pin is connected to a signal, its old output value is not copied into the signal. The proper value will only appear on the signal once the component assigns a new value.

Ansteuerung von bidirektionalen (HAL_IO) Pins

The above rule does not apply to bidirectional pins. Instead, a bidirectional pin should only be driven by the component when the component wishes to change the value. For instance, in the canonical encoder interface, the encoder component only sets the *index-enable* pin to `False` (when an index pulse is seen and the old value is `True`), but never sets it to `True`. Repeatedly driving the pin `False` might cause the other connected component to act as though another index pulse had been seen.

5.14.5. Beenden

A *halcmd unload* request for the component is delivered as a *KeyboardInterrupt* exception. When an unload request arrives, the process should either exit in a short time, or call the *.exit()* method on the component if substantial work (such as reading or writing files) must be done to complete the shutdown process.

5.14.6. HAL Python module reference

See Python HAL Interface for an overview and description of available classes and methods.

5.14.7. Konstanten

Verwenden Sie diese, um Details zu spezifizieren, und nicht den Wert, den sie enthalten.

- Pin, parameter and signal types:
  - `HAL_BOOL`
  - `HAL_REAL`
  - `HAL_SINT`
  - `HAL_UINT`
- Pin direction:
  - `HAL_IN`
  - `HAL_OUT`
- Parameter access:
  - `HAL_RO`

  - `HAL_RW`
- Message severity:
  - `MSG_NONE`
  - `MSG_ALL`
  - `MSG_DBG`
  - `MSG_ERR`
  - `MSG_INFO`
  - `MSG_WARN`

5.14.8. System-Informationen

Lesen Sie diese, um Informationen über das Echtzeitsystem zu erhalten.

- `lcnc_realtime`: See Python module for managing the realtime backend.
- `is_kernelspace`
- `is_userspace`
- `get_realtime_type()`

5.15. Kanonische Device Schnittstelle

5.15.1. Einführung

Die folgenden Abschnitte zeigen die Pins, Parameter und Funktionen, die von "kanonischen Geräten" bereitgestellt werden. Alle HAL-Gerätetreiber sollten die gleichen Pins und Parameter bereitstellen und das gleiche Verhalten implementieren.

Beachten Sie, dass nur die Felder <*io-type*> und <*specific-name*> für ein kanonisches Gerät definiert sind. Die Felder <*device-name*>, <*device-num*> und <*chan-num*> werden auf der Grundlage der Eigenschaften des realen Geräts festgelegt.

5.15.2. Digitaler Eingang

Der kanonische Digitaleingang (E/A-Typfeld: "digin") ist recht einfach.

Pins

(bool) **in**:: Zustand des Hardware-Eingangs. (bool) **in-not**:: Invertierter Zustand des Eingangs.

Parameter

Keine

Funktionen

(funct) **read**:: Lesen der Hardware und Setzen der HAL-Pins "in" und "in-not".

5.15.3. Digitaler Ausgang

Der kanonische digitale Ausgang (engl. output) (E/A-Typfeld: `digout`) ist ebenfalls sehr einfach.

Pins

(bool) **out**:: Wert, der (eventuell invertiert) an den Hardware-Ausgang geschrieben werden soll.

Parameter

(bool) **invert**:: Wenn TRUE, wird **out** vor dem Schreiben in die Hardware invertiert.

Funktionen

(funct) **write**:: Lesen Sie **out** und **invert** und stellen Sie die Hardwareausgabe entsprechend ein.

5.15.4. Analoger Eingang

Der kanonische Analogeingang (E/A-Typ: `adcin`). Dieser wird voraussichtlich für Analog-Digital-Wandler verwendet, die z. B. Spannung in einen kontinuierlichen Wertebereich umwandeln.

Pins

(real) **value**:: Der Hardware-Messwert, skaliert gemäß den Parametern **Skala** und **Offset**.
Wert = ((Eingangsmesswert, in hardwareabhängigen Einheiten) * **Skala**) - **Offset**

Parameter

(real) **scale**:: Die Eingangsspannung (oder der Strom) wird mit **Skala** multipliziert, bevor sie als **Wert** ausgegeben wird. (real) **offset**:: Dieser Wert wird nach Anwendung des Skalenmultiplikators von der Hardware-Eingangsspannung (oder dem Strom) subtrahiert. (real) **bit_weight**:: Der Wert eines niederwertigen Bits (LSB). Dies ist effektiv die Granularität des Eingangswertes. (real) **hw_offset**:: Der Wert, der am Eingang anliegt, wenn 0 Volt an den Eingangspin(s) angelegt wird.

Funktionen

(funct) **read**:: Lesen Sie die Werte dieses analogen Eingangskanals. Dies kann zum Lesen einzelner Kanäle verwendet werden, oder es können alle Kanäle gelesen werden.

5.15.5. Analoger Ausgang

Der kanonische Analogausgang (E/A-Typ: **adcout**). Dieser ist für jede Art von Hardware gedacht, die einen mehr oder weniger kontinuierlichen Wertebereich ausgeben kann. Beispiele sind Digital-Analog-

Wandler oder PWM-Generatoren.

Pins

(real) **value**:: Der zu schreibende Wert. Der tatsächliche Wert, der an die Hardware ausgegeben wird, hängt von den Parametern Skala und Offset ab. (bool) **enable**:: Wenn false, dann wird 0 an die Hardware ausgegeben, unabhängig vom **value** Pin.

Parameter

(real) **offset**:: Dies wird zu dem **Wert** (engl. value) hinzugefügt, bevor die Hardware aktualisiert wird. (real) **scale**:: Dies sollte so eingestellt werden, dass eine Eingabe von 1 am **value**-Pin dazu führt, dass der Analogausgangspin 1 Volt anzeigt. (real) **high_limit** (optional):: Wenn bei der Berechnung des an die Hardware auszugebenden Wertes **value**
offset größer ist als **high_limit**, wird stattdessen **high_limit** verwendet. (real) **low_limit** (optional):: Wenn bei der Berechnung des an die Hardware auszugebenden Wertes **value**
offset kleiner als **low_limit** ist, wird stattdessen **low_limit** verwendet. (real) **bit_weight** (optional):: Der Wert des niedrigstwertigen Bits (LSB) in Volt (oder mA bei Stromausgängen). (real) **hw_offset** (optional):: Die tatsächliche Spannung (oder Stromstärke), die ausgegeben wird, wenn 0 in die Hardware geschrieben wird.

Funktionen

(funct) **write**:: Dies bewirkt, dass der berechnete Wert an die Hardware ausgegeben wird. Wenn enable false ist, wird 0 ausgegeben, unabhängig von **value**, **scale** und **offset**. Die Bedeutung von "0" ist von der Hardware abhängig. Bei einem bipolaren 12-Bit-A/D kann es z. B. erforderlich sein, 0x1FF (mittlere Skala) an den D/A zu schreiben, um 0 Volt vom Hardware-Pin zu erhalten. Wenn enable true ist, werden Skala, Offset und Wert gelesen und an den ADC ausgegeben (**scale** (Skala) * **value** (Wert)) + **offset**. Wenn enable false ist, dann wird 0 ausgegeben.

[1] CodeAddr and Arg fields were used during development and should probably disappear.

[2] Wenn der Eingang eine Position ist, bedeutet dies, dass die *Position* begrenzt ist.

[3] Wenn der Eingang eine Position ist, bedeutet dies, dass "Position" und "Geschwindigkeit" begrenzt sind.

[4] Wenn die Eingabe eine Position ist, bedeutet dies, dass "Position", "Geschwindigkeit" und "Beschleunigung" begrenzt sind

[5] Vor Version 2.1 waren Frequenz, Amplitude und Offset Parameter. Sie wurden in Pins geändert, um die Steuerung durch andere Komponenten zu ermöglichen.

Chapter 6. Hardware-Treiber

6.1. Parallelport-Treiber

Die Komponente hal_parport ist ein Treiber für den traditionellen PC-Parallelport. Der Port hat insgesamt 17 physikalische Pins. Die ursprüngliche parallele Schnittstelle teilte diese Pins in drei Gruppen ein: Daten, Steuerung und Status. Die Datengruppe besteht aus 8 Ausgangspins, die Steuergruppe besteht aus 4 Pins und die Statusgruppe besteht aus 5 Eingangspins.

In den frühen 1990er Jahren wurde die bidirektionale parallele Schnittstelle eingeführt, die es ermöglicht, die Datengruppe für die Ausgabe oder Eingabe zu verwenden. Der HAL-Treiber unterstützt den bidirektionalen Anschluss und ermöglicht es dem Benutzer, die Datengruppe entweder als Eingang oder als Ausgang einzustellen. Ist ein Port als Ausgang konfiguriert, bietet er insgesamt 12 Ausgänge und 5 Eingänge. Wenn er als "in" konfiguriert ist, bietet er 4 Ausgänge und 13 Eingänge.

Bei einigen parallelen Anschlüssen sind die Pins der Steuergruppe offene Kollektoren, die auch durch ein externes Gate auf "low" gesetzt werden können. Auf einer Karte mit Open-Collector-Steuerpins. Bei einer Konfiguration als *x* stehen 8 Ausgänge und 9 Eingänge zur Verfügung.

Bei einigen parallelen Anschlüssen verfügt die Steuergruppe über Push-Pull-Treiber und kann nicht als Eingang verwendet werden.

NOTE

HAL und Open Collectors

HAL kann nicht automatisch feststellen, ob die bidirektionalen *x*-Modus-Pins tatsächlich offene Kollektoren (OC) sind. Wenn dies nicht der Fall ist, können sie nicht als Eingänge verwendet werden, und der Versuch, sie von einer externen Quelle auf LOW zu setzen, kann die Hardware beschädigen.

Um festzustellen, ob Ihr Port *open collector* Pins hat, laden Sie hal_parport im *x* Modus. Wenn kein Gerät angeschlossen ist, sollte HAL den Pin als TRUE lesen. Legen Sie dann einen 470 Ω-Widerstand von einem der Steuerpins auf GND. Wenn die resultierende Spannung am Steuerpin nahe bei 0 V liegt und HAL den Pin nun als FALSE liest, dann haben Sie einen OC-Port. Wenn die resultierende Spannung weit von 0 V entfernt ist oder HAL den Pin nicht als FALSE liest, dann kann Ihr Port nicht im *x*-Modus verwendet werden.

Die externe Hardware, welche die Steuerpins ansteuert, sollte ebenfalls Open-Collector-Gates verwenden, z. B. 74LS05.

Bei einigen Computern können die BIOS-Einstellungen beeinflussen, ob der *x*-Modus verwendet werden kann. Der *SPP*-Modus funktioniert am ehesten.

Andere Kombinationen werden nicht unterstützt, und ein Anschluss kann nach der Installation des Treibers nicht mehr von Eingang auf Ausgang umgestellt werden.

Der parport-Treiber kann bis zu 8 Ports steuern (definiert durch MAX_PORTS in hal_parport.c). Die Ports werden bei Null beginnend nummeriert.

6.1.1. Laden

Der hal_parport Treiber ist eine Echtzeitkomponente und muss daher mit *loadrt* in den Echtzeit-Thread geladen werden. Der Konfigurationsstring beschreibt die zu verwendenden parallelen Ports und (optional) deren Typen. Wenn der Konfigurationsstring nicht mindestens einen Port beschreibt, ist dies ein Fehler.

```
loadrt hal_parport cfg="Anschluss [Typ] [Anschluss [Typ] ...]"
```

Angabe des Ports

Zahlen unter 16 beziehen sich auf parallele Ports, die vom System erkannt werden. Dies ist der einfachste Weg, den `hal_parport`-Treiber zu konfigurieren und arbeitet mit dem Linux parport_pc-Treiber zusammen, wenn dieser geladen ist. Ein Port von 0 ist der erste vom System erkannte parallele Port, 1 ist der nächste und so weiter.

Grundkonfiguration

Dies wird den ersten parallelen Port verwenden, den Linux erkennt:

```
loadrt hal_parport cfg="0"
```

Verwenden der Portadresse

Stattdessen kann die Anschlussadresse in Hexadezimalschreibweise mit dem Präfix "0x" angegeben werden.

Die Konfigurationszeichenfolge stellt die hexadezimale Adresse des Anschlusses dar, optional gefolgt von einer Richtung, die für jeden Anschluss wiederholt wird. Die Richtungen sind *in*, *out* oder *x* und bestimmen die Richtung der physischen Pins 2 bis 9 des D-Sub 25-Anschlusses. Ist die Richtung nicht angegeben, so wird die Datengruppe standardmäßig als Ausgang konfiguriert. Zum Beispiel:

Befehl zum Laden des Echtzeitmoduls hal_partport mit dem Zusatz <config-string> zur Angabe des Ports, an dem die Parallel-Port-Karte erwartet wird.

```
loadrt hal_parport cfg="0x278 0x378 in 0x20A0 out"
```

Dieses Beispiel installiert die Treiber für einen Anschluss 0x0278, mit den Pins 2 bis 9 als Ausgänge (standardmäßig, da weder *in* noch *out* angegeben ist), einen Anschluss 0x0378, mit den Pins 2 bis 9 als Eingänge und einen Anschluss 0x20A0, mit den Pins 2 bis 9 explizit als Ausgänge angegeben. Beachten Sie, dass Sie die Basisadresse der parallelen Ports kennen müssen, um die Treiber korrekt zu konfigurieren. Bei ISA-Bus-Ports ist dies in der Regel kein Problem, da die Ports fast immer eine bekannte Adresse haben, wie z. B. 0x278 oder 0x378, die normalerweise im BIOS konfiguriert werden. Die Adressen von PCI-Bus-Karten werden normalerweise mit `lspci -v` in einer *I/O-Ports*-Zeile oder in einer Kernel-Meldung nach der Ausführung von `sudo modprobe -a parport_pc` gefunden. Es gibt keine Standardadresse, wenn also `<config-string>` nicht mindestens eine Adresse enthält, ist das ein Fehler.

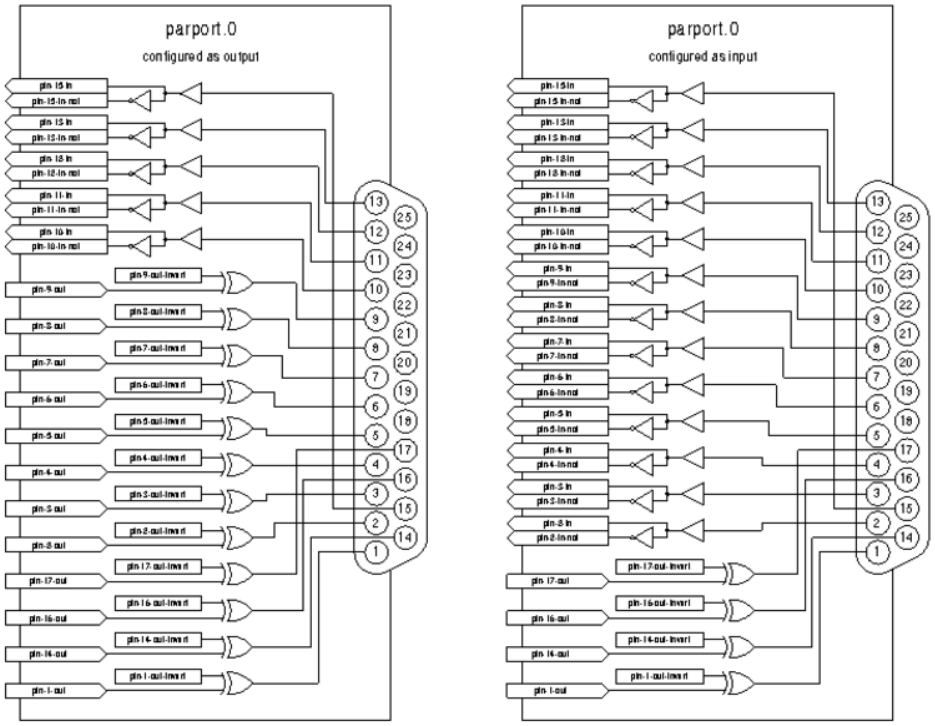


Figure 101. Parport-Blockdiagramm

Typ

Für jede parallele Schnittstelle, die vom hal_parport Treiber verwaltet wird, kann optional ein *Typ* angegeben werden. Der Typ ist einer von *in*, *out*, *epp* oder *x*.

Table 14. Parallele Port-Richtung

Pin	in	out/epp	x
1	out	out	in
2	in	out	out
3	in	out	out
4	in	out	out
5	in	out	out
6	in	out	out
7	in	out	out
8	in	out	out
9	in	out	out
10	in	in	in
11	in	in	in
12	in	in	in
13	in	in	in

Pin	in	out/epp	x
14	out	out	in
15	in	in	in
16	out	out	in
17	out	out	in

Wenn der Typ nicht angegeben wird, ist der Standardwert "out".

Ein Typ *epp* ist derselbe wie *out*, aber der hal_parport Treiber fordert den Port auf, in den EPP-Modus zu wechseln. Der hal_parport-Treiber benutzt **nicht** das EPP-Busprotokoll, aber auf einigen Systemen ändert der EPP-Modus die elektrischen Eigenschaften des Ports auf eine Weise, die einige marginale Hardware besser funktionieren lässt. Es ist bekannt, dass die Ladungspumpe des Gecko G540 dies bei einigen parallelen Anschlüssen erfordert.

Siehe den obigen Hinweis zum Modus *x*.

Beispiel mit zwei parallelen Anschlüssen

Dadurch werden zwei vom System erkannte parallele Schnittstellen aktiviert, die erste im Ausgabemodus und die zweite im Eingabemodus:

```
loadrt hal_parport cfg="0 out 1 in"
```

Parallelport Lesen und Schreiben (engl. R/W-Functions)

Sie müssen LinuxCNC auch anweisen, die Funktionen *Lesen* und *Schreiben* auszuführen.

```
addf parport.0.read base-thread
addf parport.0.write base-thread
```

6.1.2. PCI-Port-Adresse

Eine gute PCI-Parport-Karte wird mit dem Netmos 9815-Chipsatz hergestellt. Sie hat gute +5 V-Signale und kann mit einem oder zwei Anschlüssen geliefert werden.

Um die E/A-Adressen für PCI-Karten zu finden, öffnen Sie ein Terminalfenster und verwenden Sie den Befehl list pci:

```
lspci -v
```

Suchen Sie nach dem Eintrag mit "Netmos". Beispiel einer 2-Port-Karte:

```
0000:01:0a.0 Communication controller: \
      Netmos Technology PCI 9815 Multi-I/O Controller (rev 01)
Subsystem: LSI Logic / Symbios Logic 2POS (2 port parallel adapter)
Flags: medium devsel, IRQ 5
I/O ports at b800 [size=8]
```

```
I/O ports at bc00 [size=8]
I/O ports at c000 [size=8]
I/O ports at c400 [size=8]
I/O ports at c800 [size=8]
I/O ports at cc00 [size=16]
```

Beim Experimentieren habe ich festgestellt, dass der erste Anschluss (der Anschluss auf der Karte) die dritte Adresse (c000) und der zweite Anschluss (derjenige, der mit einem Flachbandkabel angeschlossen wird) die erste Adresse (b800) verwendet. Das folgende Beispiel zeigt den Onboard-Parallelport und einen PCI-Parallelport, der die Standard-Ausgangsrichtung verwendet.

```
loadrt hal_parport cfg="0x378 0xc000"
```

Bitte beachten Sie, dass Ihre Werte abweichen können. Netmos-Karten sind Plug-N-Play und könnten ihre Einstellungen ändern, je nachdem, in welchem Steckplatz sie steckt, so dass, wenn Sie gerne "unter der Haube" Dinge neu anordnen, Sie bitte darauf achten, diese Werte zu überprüfen, bevor Sie LinuxCNC starten.

6.1.3. Pins

- `parport.<p>.pin-__<n>__-out` (bool) Drives a physical output pin.
- `parport.<p>.pin-__<n>__-in` (bool) Tracks a physical input pin.
- `parport.<p>.pin-__<n>__-in-not` (bool) Tracks a physical input pin, but inverted.

Für jeden Pin ist <*p*> die Anschlussnummer und <*n*> die physische Pin-Nummer im 25-poligen D-Shell-Stecker.

Für jeden physischen Ausgangspin legt der Treiber einen einzelnen HAL-Pin an, z. B.: `parport.0.pin-14-out`.

Für jeden physischen Eingangspin erstellt der Treiber zwei HAL-Pins, zum Beispiel: `parport.0.pin-12-in` und `parport.0.pin-12-in-not`.

Der HAL-Pin `-in` ist TRUE, wenn der physikalische Pin high ist, und FALSE, wenn der physikalische Pin low ist. Der `-in-not`-HAL-Pin ist invertiert und ist FALSE, wenn der physikalische Pin high ist.

6.1.4. Parameter

- `parport.__<p>__.pin-__<n>__-out-invert` (bool) Inverts an output pin.
- `parport.__<p>__.pin-__<n>__-out-reset` (bool) (only for `-out` pins) TRUE if this pin should be reset when the `-reset` function is executed.
- `parport.__<p>__.reset-time` (UINT) The time (in nanoseconds) between a pin is set by `-write` and reset by the `-reset` function if it is enabled.

Der Parameter `-invert` bestimmt, ob ein Ausgangspin aktiv high oder aktiv low ist. Wenn `-invert` FALSE ist, wird der physikalische Pin durch das Setzen des HAL `-out`-Pins TRUE high und durch FALSE low. Wenn `-invert` TRUE ist, dann wird der physikalische Pin durch das Setzen des HAL `-out`-Pins

TRUE auf low gesetzt.

6.1.5. Funktionen

- `parport.`__<p>__.read` (funct) Liest die physikalischen Eingangspins von Port Nummer <*p*> und aktualisiert die HAL `-in` und `-in-not` Pins.
- `parport.read-all` (funct) liest physikalische Eingabepins aller Ports und aktualisiert HAL `-in` und `-in-not` Pins.
- `parport.`__<p>__.write` (funct) Liest HAL `-out` Pins von Port Nummer <*p*> und aktualisiert die physikalischen Ausgangspins dieses Ports.
- `parport.write-all` (funct) liest HAL `-out` Pins aller Ports und aktualisiert alle physikalischen Ausgangspins.
- `parport.`__<p>__.reset` (funct) Wartet, bis `reset-time` seit dem zugehörigen `write` verstrichen ist, und setzt dann die Pins auf die durch `-out-invert` und `-out-invert` angegebenen Werte zurück. `reset` muss später im selben Thread wie `write` erfolgen. Wenn `reset` TRUE ist, dann setzt die Funktion `reset` den Pin auf den Wert von `-out-invert`. Dies kann in Verbindung mit stepgen's *doublefreq* verwendet werden, um einen Schritt pro Periode zu erzeugen. Der stepgen stepspace für diesen Pin muss auf 0 gesetzt werden, um doublefreq zu aktivieren.

Die einzelnen Funktionen sind für Situationen vorgesehen, in denen ein Anschluss in einem sehr schnellen Thread aktualisiert werden muss, während andere Anschlüsse in einem langsameren Thread aktualisiert werden können, um CPU-Zeit zu sparen. Es ist wahrscheinlich keine gute Idee, gleichzeitig eine `-all` Funktion und eine individuelle Funktion zu verwenden.

6.1.6. Häufige Probleme

Wenn das Laden des Moduls berichtet

```
insmod: error inserting '/home/jepler/emc2/rtlib/hal_parport.ko':
-1 Device or resource busy
```

dann stellen Sie sicher, dass das Standard-Kernelmodul *parport_pc* nicht geladen ist Fußnote:[In den LinuxCNC-Paketen für Ubuntu verhindert die Datei /etc/modprobe.d/emc2 im Allgemeinen, dass *parport_pc* automatisch geladen wird.] und dass kein anderes Gerät im System die I/O-Ports beansprucht hat.

Wenn das Modul geladen wird, aber nicht zu funktionieren scheint, ist die Anschlussadresse falsch.

6.1.7. DoubleStep verwenden

Um DoubleStep an der parallelen Schnittstelle einzurichten, müssen Sie die Funktion parport.n.reset nach parport.n.write hinzufügen und den Schrittabstand auf 0 und die gewünschte Reset-Zeit konfigurieren. So kann der Schritt bei jeder Periode im HAL aktiviert und dann von parport ausgeschaltet werden, nachdem er für die durch `parport.`__n__.reset-time` festgelegte Zeit aktiviert wurde.

Zum Beispiel:

```
loadrt hal_parport cfg="0x378 out"
setp parport.0.reset-time 5000
loadrt stepgen step_type=0,0,0
addf parport.0.read base-thread
addf stepgen.make-pulses base-thread
addf parport.0.write base-thread
addf parport.0.reset base-thread
addf stepgen.capture-position servo-thread
...
setp stepgen.0.steplen 1
setp stepgen.0.stepspace 0
```

Weitere Informationen zu DoubleStep finden Sie im wiki.

6.1.8. probe_parport

In heutigen PCs erfordern parallele Schnittstellen möglicherweise eine Plug-and-Play-Konfiguration (PNP), bevor sie verwendet werden können. Das Kernelmodul *probe_parport* konfiguriert alle vorhandenen PNP-Ports. Es muss vor *hal_parport* geladen werden. Auf Rechnern ohne PNP-Port kann es geladen werden, hat aber keine Wirkung.

Installation von probe_parport

Wenn das Kernelmodul `parport_pc` mit dem Befehl geladen wird:

```
sudo modprobe -a parport_pc; sudo rmmod parport_pc
```

Der Linux-Kernel gibt eine Meldung ähnlich der folgenden aus:

```
parport: PnPBIOS parport erkannt.
```

Dann wird die Verwendung dieses Moduls wahrscheinlich notwendig sein.

Schließlich sollten die HAL-Parport-Komponenten geladen werden:

```
loadrt probe_parport
loadrt hal_parport ...
```

6.2. General Mechatronics Treiber

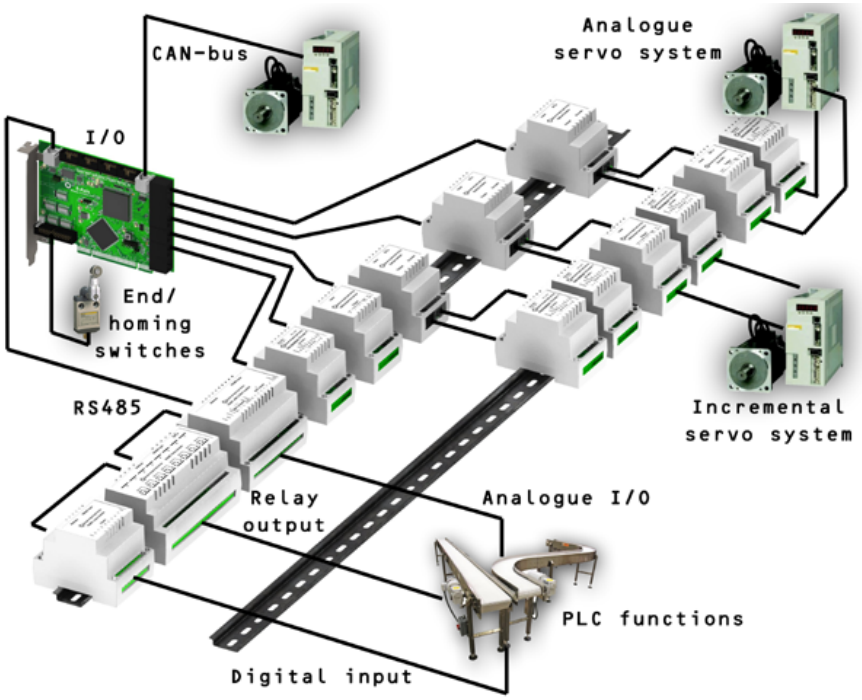
General Mechatronics GM6-PCI-Kartenbasiertes Bewegungssteuerungssystem (engl. Motion-Control-System)

Eine ausführliche Beschreibung finden Sie im Systemintegrationshandbuch.

Die GM6-PCI-Bewegungssteuerungskarte basiert auf einem FPGA und einem PCI-Bridge-Interface-ASIC.

Eine kleine automatisierte Fertigungszelle kann mit einem kurzen Systemintegrationsverfahren gesteuert werden. Die folgende Abbildung zeigt den typischen Anschluss von Geräten im Zusammenhang mit dem Steuerungssystem:

- Er kann bis zu sechs Achsen steuern, jede davon kann eine Schrittmotor- oder CAN-Bus-Schnittstelle oder ein analoger Servo sein.
- GPIO: Vier bzw. acht E/A-Pins sind auf Standard-Flachkabelsteckern untergebracht.
- RS485 E/A-Erweiterungsmodule: Der RS485-Bus wurde für den Anschluss von kompakten Erweiterungsmodulen für die DIN-Schienenmontage entwickelt. Ein 8-Kanal-Digitaleingang, ein 8-Kanal-Relaisausgang und ein analoges I/O-Modul (4x +/-10 Volt Ausgang und 8x +/-5 Volt Eingang) sind jetzt verfügbar. Insgesamt können bis zu 16 Module an den Bus angeschlossen werden.
- 20 optisch isolierte Eingangspins: Sechs mal drei für den direkten Anschluss von zwei Endschaltern und einem Referenzierungssensor für jedes Gelenk. Und zusätzlich zwei optisch isolierte Notaus-Eingänge.



Installation:

```
loadrt hal_gm
```

Während des Ladens (oder versuchten Ladens) gibt der Treiber einige nützliche Debugging-Meldungen in das Kernel-Protokoll aus, die mit dmesg eingesehen werden können.

Es können bis zu 3 Karten in einem System verwendet werden.

Die folgenden Anschlüsse befinden sich auf der GM6-PCI-Karte:

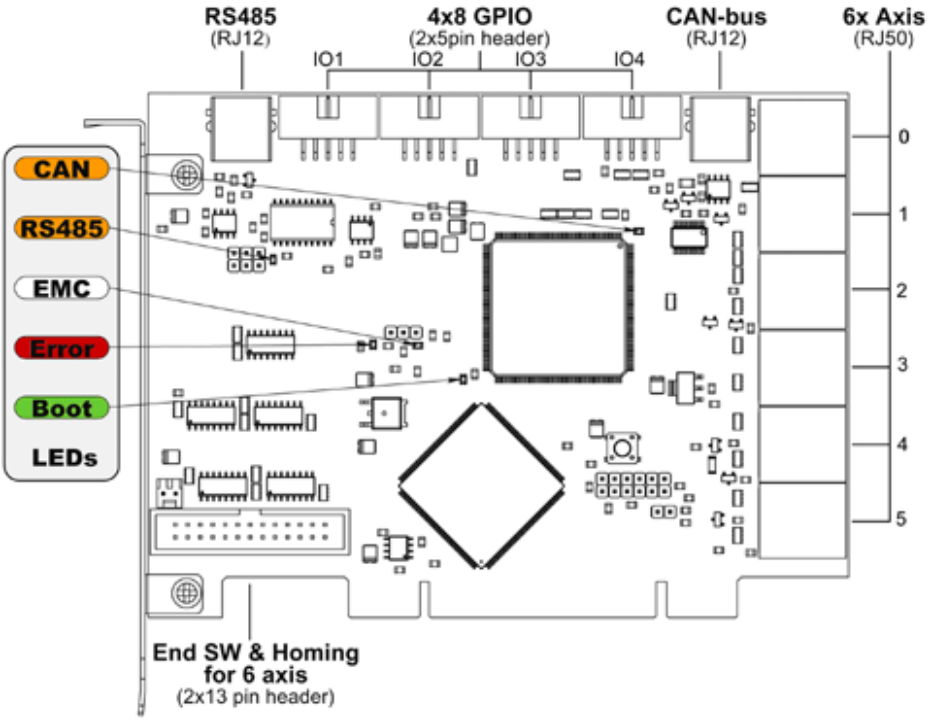


Figure 102. GM6-PCI-Kartenanschlüsse und LEDs

6.2.1. I/O-Anschlüsse

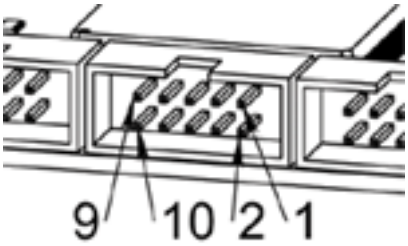


Figure 103. Pin-Nummerierung der GPIO-Anschlüsse

Table 15. Belegung der GPIO-Anschlüsse

9	7	5	3	1
IOx/7	IOx/5	IOx/3	IOx/1	VCC

10	8	6	4	2
GND	IOx/6	IOx/4	IOx/2	IOx/0

Jeder Pin kann als digitaler Eingang oder Ausgang konfiguriert werden. Die GM6-PCI-Bewegungssteuerungskarte verfügt über 4 GPIO-Anschlüsse (General Purpose I/O) mit jeweils acht konfigurierbaren E/A. Jeder GPIO-Pin und Parametername beginnt wie folgt:

```
gm.<card_no>.gpio.<gpio_con_no>
```

wobei <*gpio_con_no*> zwischen 0 und 3 liegt.

Zustand des ersten Pins des ersten GPIO-Anschlusses auf der GM6-PCI-Karte an.

```
gm.0.gpio.0.in-0
```

HAL Pins werden aktualisiert durch Funktion

```
gm.<card_no>.read
```

Pins

Table 16. GPIO-Pins

Pins	Typ und Richtung	Pin-Beschreibung
`.in-<0-7>`	(bool, Out)	Input-Pin (wörtlich Eingabe-Pin)
`.in-not-<0-7>`	(bool, Out)	Negierter Eingangspin
`.out-<0-7>`	(bool, In)	Ausgangspin. Wird nur verwendet, wenn GPIO auf Ausgang eingestellt ist.

Parameter

Table 17. GPIO-Parameter

Pins	Typ und Richtung	Parameterbeschreibung
`.is-out-<0-7>`	(bool, R/W)	Bei True wird der entsprechende GPIO auf Totem-Pol-Ausgang gesetzt, andernfalls auf hochohmigen Eingang.
`.invert-out-<0-7>`	(bool, R/W)	Wenn True, wird der Wert des Pins invertiert. Wird verwendet, wenn der Pin als Ausgang konfiguriert ist.

6.2.2. Achsen-Anschlüsse

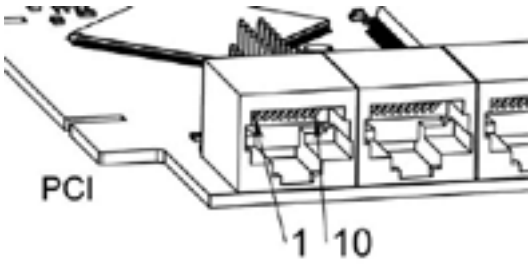


Figure 104. Pin-Nummerierung der Achsenverbinder

Table 18. Belegung der Achsanschlüsse

1	Encoder A
2	+5 Volt (PC)
3	Encoder B
4	Encoder-Index
5	Fehler
6	Stromversorgung aktiviert (engl. power enabled)
7	Schritt/gegen Uhrzeigersinn/B
8	Richtung/Uhrzeigersin/A
9	Masse (PC)
10	Serielle DAC-Leitung

Achsen-Schnittstellenmodule

Kleine, auf DIN-Schienen montierte Schnittstellenmodule ermöglichen den einfachen Anschluss verschiedener Servomodule an die Achsanschlüsse. Sieben verschiedene Systemkonfigurationen werden im Systemintegrationshandbuch vorgestellt, um typische Anwendungen zu evaluieren. Auch die detaillierte Beschreibung der Achsmodule ist im Systemintegrationshandbuch zu finden.

Zur Ermittlung der geeigneten Servoantriebsstruktur sind die Module wie im folgenden Blockschaltbild dargestellt zu verbinden:

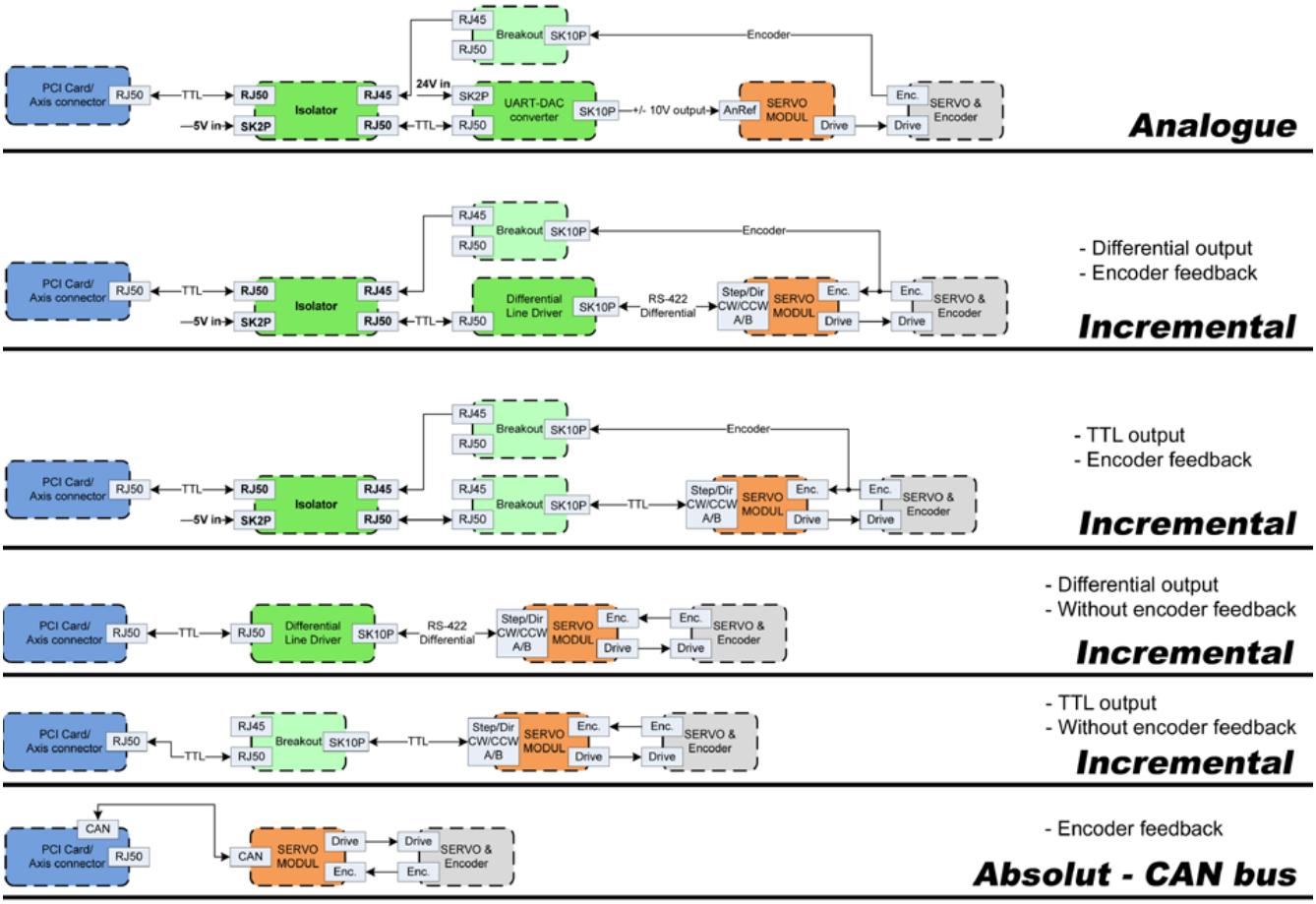


Figure 105. Servo-Achsen-Schnittstellen

Encoder

Die GM6-PCI-Bewegungssteuerungskarte verfügt über sechs Encoder-Module. Jedes Gebermodul hat drei Kanäle:

- Kanal-A
- Kanal-B
- Kanal-I (Index)

Es ist in der Lage, Quadratur-Encoder-Signale oder Schritt-/Taktsignale zu zählen. Jedes Encoder-Modul wird an die Eingänge des entsprechenden RJ50-Achsenanschlusses angeschlossen.

Jeder Encoder-Pin und Parametername beginnt wie folgt:

```
gm.<card_no>.encoder.<axis_no>
```

wobei <*axis_no*> zwischen 0 und 5 liegt. Beispielsweise bezieht sich „gm.0.encoder.0.position“ auf die Position des Encodermoduls von Achse 0.

Die GM6-PCI-Karte zählt das Gebersignal unabhängig von LinuxCNC. HAL-Pins werden nach Funktion aktualisiert:

```
gm.<card_no>.read
```

Table 19. Encoder-Pins

Pins	Typ und Richtung	Pin-Beschreibung
`.reset`	(bool, In)	Wenn True, setzt Anzahl und Position auf Null zurück.
`.rawcounts`	(sint, Out)	Die "raw counts" ist dieselbe Zählung, aber unbeeinflusst von der Rückstellung oder dem Indeximpuls.
`.counts`	(sint, Out)	Position in Encoder-Zählungen (engl. counts).
`.position`	(real, Out)	Position in skalierten Einheiten (=.counts/.position-scale).
`.index-enable`	(bool, IO)	When True, counts and position are rounded or reset (depends on index-mode) on next rising edge of channel-I. Every time position is reset because of Index, the `index-enable` pin is set to 0 and remains 0 until connected HAL pin does not set it.
`.velocity`	(real, Out)	Geschwindigkeit in skalierten Einheiten pro Sekunde. Der GM-Encoder verwendet einen Hochfrequenz-Hardware-Timer zur Messung der Zeit zwischen den Encoderimpulsen, um die Geschwindigkeit zu berechnen. Dadurch wird das Quantisierungsrauschen im Vergleich zur einfachen Differenzierung des Positionsausgangs erheblich reduziert. Wenn die gemessene Geschwindigkeit unter der geschätzten Mindestgeschwindigkeit liegt, ist der Geschwindigkeitsausgang 0.

Table 20. Encoder-Parameter

Parameter	Typ und Lesen/Schreiben	Parameterbeschreibung
.counter-mode	(bool, R/W)	Bei True zählt der Zähler jede ansteigende Flanke des Kanal-A-Eingangs in die durch Kanal-B bestimmte Richtung. Dies ist nützlich für die Zählung des Ausgangs eines Einkanal- (Nicht-Quadratur-) oder Schritt-/Differenzsignalsensors. Wenn False, zählt er im Quadraturmodus.
.index-mode	(bool, R/W)	Wenn True und .index-enabled ebenfalls True ist, werden .counts und .position gerundet (basierend auf .counts-per-rev) bei der steigenden Flanke von Kanal-I. Dies ist nützlich, um durch Rauschen verursachte Fehler bei wenigen Impulsen zu korrigieren. Im Rundungsmodus ist es wichtig, dass der Parameter .counts-per-rev korrekt eingestellt ist. Wenn .index-mode auf False und .index-enabled auf True steht, werden .counts und .position beim Kanal-I-Impuls zurückgesetzt.
.counts-per-rev	(uint, R/W)	Bestimmt, wie viele Zählungen zwischen zwei Indeximpulsen liegen. Er wird nur im runden Modus verwendet, d.h. wenn die Parameter .index-enabled und .index-mode beide auf True stehen. GM-Encoder verarbeiten das Encodersignal im 4x-Modus, d.h. im Falle eines 500 CPR Encoders sollte er auf 2000 gesetzt werden. Dieser Parameter kann leicht gemessen werden, indem man .index-enabled True und .index-mode False setzt (so dass .counts bei Kanal-I-Impuls zurücksetzt), dann die Achse von Hand bewegen und die maximale Größe des .counts-Pins im Halmeter sehen.
.index-invert	(bool, R/W)	Wenn True, tritt das Kanal-I-Ereignis (Reset oder Round) bei der fallenden Flanke des Kanal-I-Signals ein, andernfalls bei der steigenden Flanke.

Parameter	Typ und Lesen/Schreiben	Parameterbeschreibung
.min-speed-estimate	(real, R/W)	Legen Sie den minimalen gemessenen Geschwindigkeitswert fest, bei dem .velocity als ungleich Null gesetzt wird. Wird dieser Parameter zu niedrig eingestellt, dauert es sehr lange, bis die Geschwindigkeit auf Null geht, nachdem keine Encoderimpulse mehr ankommen.
.position-scale	(real, R/W)	Skala in Zählungen pro Längeneinheit. .position=.counts/.position-scale. Wenn z. B. der Positionsmaßstab 2000 ist, dann ergeben 1000 Zählungen des Encoders eine Position von 0,5 Einheiten.

Einstellung des Gebermoduls der Achse 0 für den Empfang des 500 CPR Quadraturgebersignals und Verwendung der Rückstellung zum Runden der Position.

```
setp gm.0.encoder.0.counter-mode 0 # 0: quad, 1: stepDir
setp gm.0.encoder.0.index-mode 1 # 0: reset pos at index, 1:round pos at index
setp gm.0.encoder.0.counts-per-rev 2000 # GM-Prozess-Encoder im 4x-Modus, 4x500=2000
setp gm.0.encoder.0.index-invert 0 #
setp gm.0.encoder.0.min-speed-estimate 0.1 # in Positionseinheit/s
setp gm.0.encoder.0.position-scale 20000 # 10 Geberumdrehungen bewirken, dass sich die
Maschine um eine Positionseinheit bewegt (10x2000)
```

Verbinden der Encoder Position mit LinuxCNC Gelenk-Position Feedback

```
net Xpos-fb gm.0.encoder.0.position => joint.0.motor-pos-fb
```

StepGen Modul

Die GM6-PCI-Bewegungssteuerungskarte verfügt über sechs StepGen-Module, eines für jedes Gelenk. Jedes Modul hat zwei Ausgangssignale. Es kann Schritt/Richtung-, Auf/Ab- oder Quadraturimpulse (A/B) erzeugen. Jedes StepGen-Modul wird an die Stifte des entsprechenden RJ50-Achsenanschlusses angeschlossen.

Jeder StepGen-Pin und Parametername beginnt wie folgt:

```
gm.<card_no>.stepgen.<axis_no>
```

wobei <*axis_no*> zwischen 0 und 5 liegt. Zum Beispiel bezieht sich *gm.0.stepgen.0.position-cmd* auf den Positionsbefehl des StepGen-Moduls der Achse 0 auf Karte 0.

Die GM6-PCI-Karte erzeugt unabhängig von LinuxCNC Schrittimpulse. Die HAL-Pins werden geupdatet durch die Funktion

```
gm.<card_no>.write
```

Table 21. StepGen Modul Pins

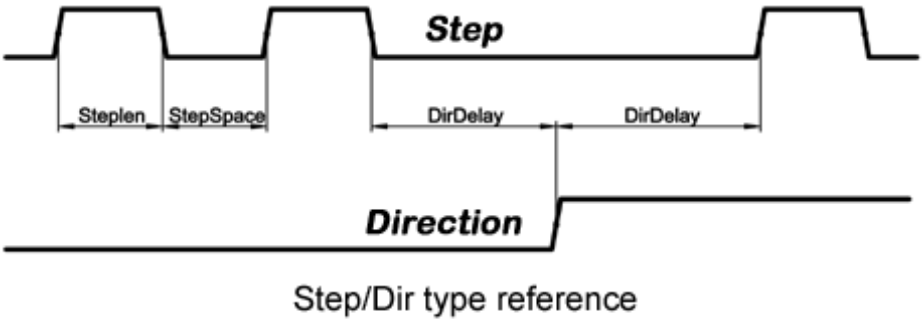
Pins	Typ und Richtung	Pin-Beschreibung
.enable	(bool, In)	StepGen erzeugt nur dann Impulse, wenn dieser Pin "true" ist.
.count-fb	(sint, Out)	Positionsrückmeldung in Zähleinheiten.
.position-fb	(real, Out)	Positionsrückmeldung in Positionseinheit.
.position-cmd	(real, In)	Die befohlene Position in Positionseinheiten. Wird nur im Positionsmodus verwendet.
.velocity-cmd	(real, In)	Geforderte Geschwindigkeit in Positionseinheiten pro Sekunde. Wird nur im Geschwindigkeitsmodus verwendet.

Table 22. StepGen-Modul-Parameter

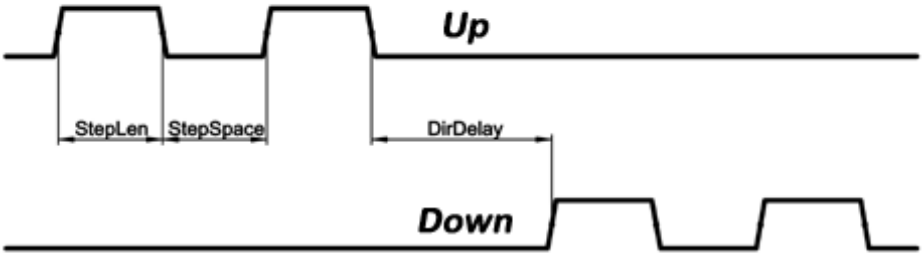
Parameter	Typ und Lesen/Schreiben	Parameterbeschreibung
.step-type	(uint, R/W)	Bei 0 erzeugt das Modul ein Step/Dir-Signal. Wenn 1, erzeugt es Auf/Ab-Schritt-Signale. Und bei 2 erzeugt es Quadratur-Ausgangssignale.
.control-type	(bool, R/W)	Wenn True, wird .velocity-cmd als Referenz verwendet und velocityvcontrol berechnet die Pulsfrequenzausgabe. Bei False wird .position-cmd als Referenz verwendet und die Lageregelung berechnet den Impulsratenausgang.
.invert-step1	(bool, R/W)	Invertieren des Ausgangs von Kanal 1 (Step-Signal im StepDir-Modus)
.invert-step2	(bool, R/W)	Invertierung des Ausgangs von Kanal 2 (Dir-Signal im StepDir-Modus)
.maxvel	(real, R/W)	Maximale Geschwindigkeit in Positionseinheiten pro Sekunde. Wenn er auf 0,0 gesetzt ist, wird der Parameter .maxvel ignoriert.
.maxaccel	(real, R/W)	Maximale Beschleunigung in Positionseinheiten pro Sekunde zum Quadrat. Wenn er auf 0,0 gesetzt ist, wird der Parameter .maxaccel ignoriert.

Parameter	Typ und Lesen/Schreiben	Parameterbeschreibung
.position-scale	(real, R/W)	Skala in Schritten pro Längeneinheit.
.steplen	(uint, R/W)	Länge des Schritt-Pulses (engl. step pulse) in Nanosekunden.
.stepspace	(uint, R/W)	Mindestzeit zwischen zwei Schrittimpulsen in Nanosekunden.
.dirdelay	(uint, R/W)	Mindestzeit zwischen Schrittimpuls und Richtungswechsel in Nanosekunden.

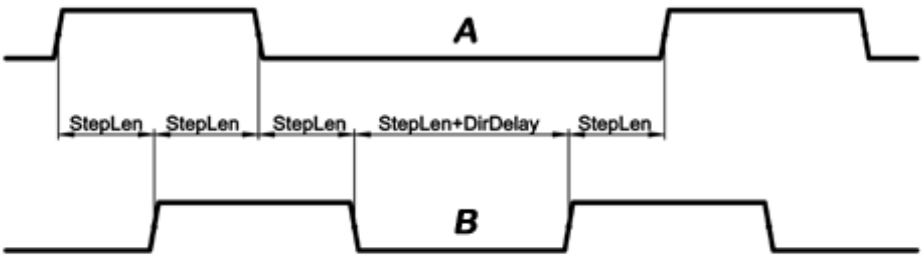
Zur Ermittlung der entsprechenden Werte siehe die nachstehenden Zeitdiagramme:



Step/Dir type reference



Up/Down count (CW/CCW) reference



Quadrant (A/B) type reference

Figure 106. Referenzsignal-Zeitdiagramme

Einstellung des StepGen-Moduls der Achse 0 zur Erzeugung von 1000 Schrittimpulsen pro Positionseinheit

```
setp gm.0.stepgen.0.step-type 0         # 0:stepDir, 1:UpDown, 2:Quad
setp gm.0.stepgen.0.control-type 0      # 0:Pos. control, 1:Vel. Control
setp gm.0.stepgen.0.invert-step1 0
setp gm.0.stepgen.0.invert-step2 0
setp gm.0.stepgen.0.maxvel 0            # do not set maxvel for step
                                        # generator, let interpolator control it.
setp gm.0.stepgen.0.maxaccel 0          # do not set max acceleration for
                                        # step generator, let interpolator control it.
setp gm.0.stepgen.0.position-scale 1000 # 1000 step/position unit
setp gm.0.stepgen.0.steplen 1000        # 1000 ns = 1 µs
setp gm.0.stepgen.0.stepspace 1000       # 1000 ns = 1 µs
setp gm.0.stepgen.0.dirdelay 2000       # 2000 ns = 2 µs
```

Verbinden Sie StepGen mit den Positionsreferenz- und Freigabepins der Achse 0

```
net Xpos-cmd joint.0.motor-pos-cmd => gm.0.stepgen.0.position-cmd
net Xen joint.0.amp-enable-out => gm.0.stepgen.0.enable
```

Freigabe- und Fehlersignale

Die GM6-PCI-Bewegungssteuerungskarte verfügt über einen HAL-Pin als Freigabeausgang und einen als Fehlereingang , die beide mit jedem RJ50-Achsenanschluss und dem CAN-Anschluss verbunden sind.

HAL Pins werden durch Funktion aktualisiert:

```
gm.<card_no>.read
```

Table 23. Aktivierungs- und Fehlersignal-Pins

Pins	Typ und Richtung	Pin-Beschreibung
gm.<card_no>.power-enable	(bool, In)	Wenn dieser Pin True ist, * und der Watch Dog Timer nicht abgelaufen ist * und es liegt kein Stromfehler vor dann werden die Power-Enable-Pins der Achsen- und CAN-Anschlüsse auf High gesetzt, ansonsten auf Low.
gm.<card_no>.power-fault	(bool, Out)	Stromausfall-Eingang.

Achsen-DAC

Die GM6-PCI-Bewegungssteuerungskarte verfügt über sechs serielle Achsen-DAC-Treibermodule, eines für jedes Gelenk. Jedes Modul wird an den Pin des entsprechenden RJ50-Achsenanschlusses angeschlossen. Jeder Achsen-DAC-Pin und Parametername beginnt wie folgt:

```
gm.<card_no>.dac.<axis_no>
```

wobei *<axis_no>* zwischen 0 und 5 liegt. Zum Beispiel bezieht sich *gm.0.dac.0.value* auf die Ausgangsspannung des DAC-Moduls der Achse 0.

HAL Pins werden durch Funktion aktualisiert:

```
gm.<card_no>.write
```

Table 24. Achsen DAC-Pins

Pins	Typ und Richtung	Pin-Beschreibung
.enable	(bool, In)	Aktiviert (engl. enables) den DAC-Ausgang. Wenn diese Freigabe auf falsch gesetzt ist, beträgt der DAC-Ausgang 0,0 V.
.value	(real, In)	Wert des DAC-Ausgangs in Volt.

Table 25. Achsen-DAC-Parameter

Parameter	Typ und Richtung	Parameterbeschreibung
.offset	(real, R/W)	Der Offset wird zu dem Wert addiert, bevor die Hardware aktualisiert wird.
.high-limit	(real, R/W)	Maximale Ausgangsspannung der Hardware in Volt.
.Untergrenze	(real, R/W)	Minimale Ausgangsspannung der Hardware in Volt.
.invert-seriell	(bool, R/W)	Die GM6-PCI-Karte kommuniziert mit der DAC-Hardware über schnelle serielle Kommunikation, um die Zeitverzögerung im Vergleich zur PWM stark zu reduzieren. Es wird empfohlen, das DAC-Modul zu isolieren, wodurch die serielle Kommunikationsleitung negiert wird. Im Falle einer Isolierung lassen Sie diesen Parameter auf dem Standardwert (0), während Sie diesen Parameter im Falle einer Nicht-Isolierung auf 1 setzen.

6.2.3. CAN-Bus-Servoverstärker

Die GM6-PCI Motion Control Karte verfügt über ein CAN-Modul zur Ansteuerung von CAN-Servoverstärkern. Die Implementierung von Protokollen auf höherer Ebene wie CANopen ist für eine

zukünftige Entwicklung. Derzeit haben die von GM hergestellten Leistungsverstärker Treiber der oberen Ebene, die Pins und Parameter nach HAL exportieren. Sie empfangen Positionsreferenzen und liefern Encoder-Feedback über den CAN-Bus.

Die Frames sind Standard-ID-Frames (11 Bit) mit einer Datenlänge von 4 Byte. Die Baudrate beträgt 1 Mbit/s. Die Positionsbefehls-IDs für Achse 0..5 sind 0x10..0x15. Die Positionsrückmelde-IDs für Achse 0..5 sind 0x20..0x25.

Diese Konfiguration kann mit der Änderung der hal_gm.c und Neukompilierung LinuxCNC geändert werden.

Jeder CAN-Pin- und Parametername beginnt wie folgt:

```
gm.<card_no>.can-gm.<axis_no>
```

where *<axis_no>* is from 0 to 5. For example, `gm.0.can-gm.0.position-fb` refers to the feedback position of axis 0 in position units.

HAL Pins werden durch Funktion aktualisiert:

```
gm.<card_no>.write
```

Pins

Table 26. CAN-Modul-Pins

Pins	Typ und Richtung	Pin-Beschreibung
`.enable`	(bool, In)	Aktivieren das Senden von Positionsreferenzen.
`.position-cmd`	(real, In)	Befohlene Position in Positionseinheiten.
`.position-fb`	(real, Out)	Rückmeldung der Position in Positionseinheiten.

Parameter

Table 27. CAN-Modul-Parameter

Parameter	Typ und Richtung	Parameterbeschreibung
`.position-scale`	(real, R/W)	Maßstab in Längeneinheiten.

6.2.4. Watchdog-Timer

Watchdog-Timer wird zurückgesetzt bei Funktion:

```
gm.<card_no>.read
```

Pins

Table 28. Watchdog-Pins

Pins	Typ und Richtung	Pin-Beschreibung
gm.<*card_no*>.watchdog-expired	(bool, Out)	Gibt an, dass der Watchdog-Zeitgeber abgelaufen ist.

Das Überschreiten des Watchdog-Timers führt dazu, dass das Power-Enable in der Hardware auf Low gesetzt wird.

Parameter

Table 29. Watchdog-Parameter

Parameter	Typ und Richtung	Parameterbeschreibung
gm.<*card_no*>.watchdog-enable	(bool, R/W)	Aktiviert den Watchdog-Timer. Es wird dringend empfohlen, den Watchdog-Timer zu aktivieren, weil er im Falle eines PC-Fehlers alle Servoverstärker durch Herunterziehen aller Freigabesignale deaktivieren kann.
gm.<*card_no*>.watchdog-timeout-ns	(uint, R/W)	Zeitintervall, in dem die Funktion gm.<*card_no*>.read ausgeführt werden muss. Die gm.<*card_no*>.read wird typischerweise dem Servo-Thread hinzugefügt, daher wird der Watch-Timeout typischerweise auf das 3-fache der Servoperiode gesetzt.

6.2.5. End-, Referenzpunkt- und Notaus-Schalter

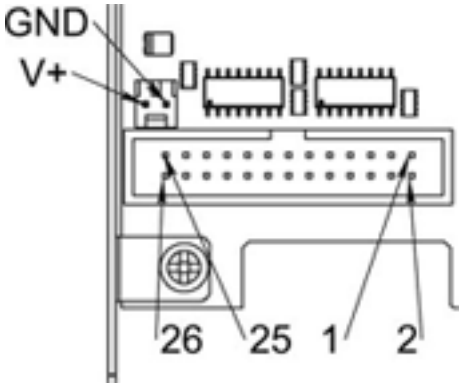


Figure 107. Pin-Nummerierung des Anschlusses für Referenzfahrt und Endschalter

Table 30. Pinbelegung des End- und Referenzschalteranschlusses

25	23	21	19	17	15	13	11	9	7	5	3	1
GND		1/End-	2/End+	2/Hom-ing	3/End-	4/End+	4/Hom-ing	5/End-	6/End+	6/Hom-ing	Notaus 2	V+ (Ext.)

26	24	22	20	18	16	14	12	10	8	6	4	2
GND		1/End+	1/Hom-ing	2/End-	3/End+	3/Hom-ing	4/End-	5/End+	5/Hom-ing	6/End-	Notaus 1	V+ (Ext.)

Die GM6-PCI-Bewegungssteuerungskarte hat zwei Endschalter- und einen Referenzschaltereingang für jedes Gelenk. Die Namen dieser Pins beginnen wie folgt:

```
gm.<card_no>.axis.<axis_no>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.axis.0.home-sw-in` indicates the state of the axis 0 home switch.

HAL Pins werden durch Funktion aktualisiert:

```
gm.<card_no>.read
```

Pins

Table 31. End- und Referenzschalter-Pins

Pins	Typ und Richtung	Pin-Beschreibung
`.home-sw-in`	(bool, Out)	Eingang des Referenzschalters
`.home-sw-in-not`	(bool, Out)	Negierter Referenzschaltereingang
`.neg-lim-sw-in`	(bool, Out)	Negativer Endschaltereingang
`.neg-lim-sw-in-not`	(bool, Out)	Negierter negativer Endschaltereingang
`.pos-lim-sw-in`	(bool, Out)	Positiver Endschaltereingang
`.pos-lim-sw-in-not`	(bool, Out)	Negierter positiver Endschaltereingang

Table 32. E-stop switch pins

Pins	Typ und Richtung	Pin-Beschreibung
`gm.0.estop.0.in`	(bool, Out)	Notaus0 Eingang

Pins	Typ und Richtung	Pin-Beschreibung
gm.0.estop.0.in-not	(bool, Out)	Negierter Notaus 0-Eingang
gm.0.estop.1.in	(bool, Out)	Notaus 1 Eingang
gm.0.estop.1.in-not	(bool, Out)	Negierter Notaus 1-Eingang

6.2.6. Status-LEDs

CAN

Farbe: Orange

- Blinken während der Datenkommunikation.
- Ein, wenn einer der Puffer voll ist - Kommunikationsfehler.
- Aus, wenn keine Datenkommunikation stattfindet.

RS485

Farbe: Orange

- Blinken während der Initialisierung von Modulen auf dem Bus
- Ein, wenn die Datenkommunikation zwischen allen initialisierten Modulen hergestellt ist.
- Aus, wenn eines der initialisierten Module aufgrund eines Fehlers ausgefallen ist.

EMC

Farbe: Weiß

- Blinken, wenn LinuxCNC läuft.
- Sonst aus.

Booten

Farbe: Grün

- Ein, wenn das System erfolgreich gebootet wurde.
- Sonst aus.

Fehler

Farbe: Rot

- Aus, wenn keine Störung im System vorliegt.
- Blinkt, wenn ein PCI-Kommunikationsfehler vorliegt.
- Ein, wenn der Watchdog-Timer übergelaufen ist.

6.2.7. RS485 E/A-Erweiterungsmodule

Diese Module wurden für die Erweiterung der E/A- und Funktionsfähigkeit entlang einer RS485-Linie der GM6-PCI Motion Control Karte entwickelt.

Verfügbare Modultypen:

- 8-Kanal-Relaisausgangsmodul - bietet acht NO-NC-Relaisausgänge an einem dreipoligen Klemmenanschluss für jeden Kanal.
- 8-Kanal-Digitaleingangsmodul - bietet acht optisch isolierte digitale Eingangsstifte.
- 8-Kanal-ADC- und 4-Kanal-DAC-Modul - bietet vier Digital-Analog-Wandler-Ausgänge und acht Analog-Digital-Eingänge. Auch dieses Modul ist von der GM6-PCI-Karte optisch isoliert.

Automatische Knotenerkennung

Jeder an den Bus angeschlossene Knoten wurde von der GM6-PCI-Karte automatisch erkannt. Beim Start von LinuxCNC exportiert der Treiber automatisch Pins und Parameter aller verfügbaren Module.

Fehlerbehandlung

Wenn ein Modul nicht regelmäßig antwortet, fährt die GM6-PCI-Karte das Modul herunter. Wenn ein Modul mit Ausgang nicht regelmäßig Daten mit korrektem CRC erhält, schaltet das Modul in den Fehlerzustand (grüne LED blinkt), und schaltet alle Ausgänge in den Fehlerzustand.

Verbinden der Knoten

Die Module auf dem Bus müssen in serieller Topologie angeschlossen werden, mit Abschlusswiderständen am Ende. Der Anfang der Topologie ist die PCI-Karte, und das Ende ist das letzte Modul.

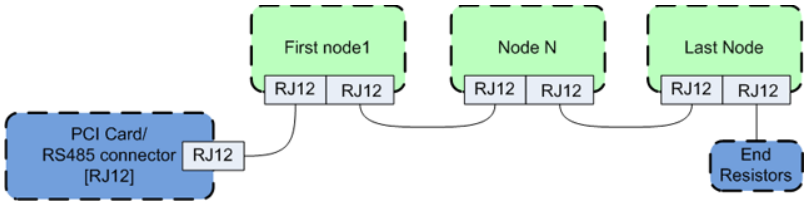


Figure 108. Anschließen der RS485-Knoten an die GM6-PCI-Karte

Adressierung

Jeder Knoten am Bus hat eine eindeutige 4-Bit-Adresse, die mit einem roten DIP-Schalter eingestellt werden kann.

Status-LED

Eine grüne LED zeigt den Status des Moduls an:

- Blinkt, wenn das Modul nur mit Strom versorgt, aber noch nicht erkannt wird, oder wenn das Modul

fallengelassen wird.

- Aus, während der Identifizierung (Computer ist eingeschaltet, aber LinuxCNC nicht gestartet)
- Ein, wenn es kontinuierlich kommuniziert.

Relais-Ausgangsmodul

Informationen zu Pinbelegung, Anschluss und elektrischen Eigenschaften des Moduls finden Sie im Systemintegrationshandbuch.

Alle Pins und Parameter werden durch die folgende Funktion aktualisiert:

```
gm.<card_no>.rs485
```

Es sollte dem Servo-Thread oder einem anderen Thread mit größerer Periode hinzugefügt werden, um eine CPU-Überlastung zu vermeiden. Jeder RS485-Modul-Pin und Parametername beginnt wie folgt:

```
gm.<card_no>.rs485.<module ID>
```

wobei <*Modul-ID*> zwischen 00 und 15 liegt.

Table 33. Pins des Relaisausgangsmoduls

Pins	Typ und Richtung	Pin-Beschreibung
.relay-<0-7>	(bool, In)	Ausgangspin für Relais

Table 34. Parameter des Relaisausgangsmoduls

Parameter	Typ und Richtung	Parameterbeschreibung
.invert-relay-<0-7>	(bool, R/W)	Relais-Ausgangsstift negieren

HAL-Beispiel

```
  gm.0.rs485.00.relay-0   # First relay of the node.
# gm.0                   # Identifies the first GM6-PCI motion control card (PCI card
address = 0)
#     .rs485.0           # Selects node with address 0 on the RS485 bus
#             .relay-0   # Selects the first relay
```

Digitales Eingangsmodul

Informationen zu Pinbelegung, Anschluss und elektrischen Eigenschaften des Moduls finden Sie im Systemintegrationshandbuch.

Alle Pins und Parameter werden durch die folgende Funktion aktualisiert:

```
gm.<card_no>.rs485
```

Es sollte dem Servo-Thread oder einem anderen Thread mit größerer Periode hinzugefügt werden, um eine CPU-Überlastung zu vermeiden. Jeder RS485-Modul-Pin und Parametername beginnt wie folgt:

```
gm.<card_no>.rs485.<module ID>
```

wobei <*Modul-ID*> zwischen 00 und 15 liegt.

Table 35. Pins des digitalen Eingangs-/Ausgangsmoduls

Pins	Typ und Richtung	Pin-Beschreibung
.in-<0-7>	(bool, Out)	Input-Pin (wörtlich Eingabe-Pin)
.in-not-<0-7>	(bool, Out)	Negierter Eingangspin

HAL-Beispiel

```
  gm.0.rs485.00.in-0   # First input of the node.
# gm.0                # Identifies the first GM6-PCI motion control card (PCI card
address = 0)
#     .rs485.0        # Selects node with address 0 on the RS485 bus
#             .in-0   # Selects the first digital input module
```

DAC & ADC-Modul

Informationen zu Pinbelegung, Anschluss und elektrischen Eigenschaften des Moduls finden Sie im Systemintegrationshandbuch.

Alle Pins und Parameter werden durch die folgende Funktion aktualisiert:

```
gm.<card_no>.rs485
```

Es sollte dem Servo-Thread oder einem anderen Thread mit größerer Periode hinzugefügt werden, um eine CPU-Überlastung zu vermeiden. Jeder RS485-Modul-Pin und Parametername beginnt wie folgt:

```
gm.<card_no>.rs485.<module ID>
```

wobei <*Modul-ID*> zwischen 00 und 15 liegt.

Table 36. DAC- & ADC-Modul-Pins

Pins	Typ und Richtung	Pin-Beschreibung
.adc-<0-7>	(real, Out)	Wert des ADC-Eingangs in Volt.

Pins	Typ und Richtung	Pin-Beschreibung
.dac-enable-<0-3>	(bool, In)	Aktivieren Sie den DAC-Ausgang. Wenn enable false ist, wird der DAC-Ausgang auf 0,0 V gesetzt.
.dac-<0-3>	(real, In)	Wert des DAC-Ausgangs in Volt.

Table 37. Parameter des DAC- & ADC-Moduls

Parameter	Typ und Richtung	Parameterbeschreibung
.adc-scale-<0-7>	(real, R/W)	Die Eingangsspannung wird mit der Skalierung multipliziert, bevor sie an den .adc-Pin ausgegeben wird.
.adc-offset-<0-7>	(real, R/W)	Der Offset wird von der Hardware-Eingangsspannung subtrahiert, nachdem der Skalenmultiplikator angewendet wurde.
.dac-offset-<0-3>	(real, R/W)	Der Offset wird zu dem Wert addiert, bevor die Hardware aktualisiert wird.
.dac-high-limit-<0-3>	(real, R/W)	Maximale Ausgangsspannung der Hardware in Volt.
.dac-low-limit-<0-3>	(real, R/W)	Minimale Ausgangsspannung der Hardware in Volt.

HAL-Beispiel

```
  gm.0.rs485.00.adc-0   # First analogue channel of the node.
# gm.0                  # Identifies the first GM6-PCI motion control card (PCI card
address = 0)
#     .rs485.0          # Selects node with address 0 on the RS485 bus
#             .adc-0    # Selects the first analogue input of the module
```

Teach Pendant Modul

Informationen zu Pinbelegung, Anschluss und elektrischen Eigenschaften des Moduls finden Sie im Systemintegrationshandbuch.

Alle Pins und Parameter werden durch die folgende Funktion aktualisiert:

```
gm.<card_no>.rs485
```

Es sollte dem Servo-Thread oder einem anderen Thread mit größerer Periode hinzugefügt werden, um eine CPU-Überlastung zu vermeiden. Jeder RS485-Modul-Pin und Parametername beginnt wie folgt:

```
gm.<card_no>.rs485.<module ID>
```

wobei <*Modul-ID*> zwischen 00 und 15 liegt. Beachten Sie, dass sie beim Teach-Pendant-Modul nicht

geändert werden kann und auf Null vorprogrammiert ist. Auf Anfrage kann das Modul mit einer anderen, von der Firmware vorprogrammierten ID geliefert werden.

Table 38. Pins des Teach-Pendant-Moduls

Pins	Typ und Richtung	Pin-Beschreibung
.adc-<0-5>	(real, Out)	Wert des ADC-Eingangs in Volt.
.enc-reset	(bool, In)	Wenn True, setzt Anzahl und Position auf Null zurück.
.enc-counts	(sint, Out)	Position in Encoder-Zählungen (engl. counts).
.enc-rawcounts	(sint, Out)	Die rohen Zähldaten (engl. raw counts) ist die Zählung, die durch das Zurücksetzen nicht beeinflusst wird.
.enc-position	(real, Out)	Position in skalierten Einheiten (=.enc-counts/.enc-position-scale).
.in-<0-7>	(bool, Out)	Input-Pin (wörtlich Eingabe-Pin)
.in-not-<0-7>	(bool, Out)	Negierter Eingangspin

Table 39. Parameter des Teach Pendant Moduls

Parameter	Typ und Richtung	Parameterbeschreibung
.adc-scale-<0-5>	(real, R/W)	Die Eingangsspannung wird mit der Skalierung multipliziert, bevor sie an den .adc-Pin ausgegeben wird.
.adc-offset-<0-5>	(real, R/W)	Der Offset wird von der Hardware-Eingangsspannung subtrahiert, nachdem der Skalenmultiplikator angewendet wurde.
.enc-position-scale	(real, R/W)	Maßstab in Längeneinheiten.

HAL-Beispiel

```
  gm.0.rs485.00.adc-0  # First analogue channel of the node.
# gm.0                # Identifies the first GM6-PCI motion control card (PCI card
address = 0)
#     .rs485.0        # Selects node with address 0 on the RS485 bus
#             .adc-0  # Selects the first analogue input of the module
```

6.2.8. Errata

GM6-PCI-Karte Errata

Die Revisionsnummer in diesem Abschnitt bezieht sich auf die Revision des GM6-PCI-Kartengeräts.

Rev. 1.2

- Fehler: Die PCI-Karte bootet nicht, wenn der Schalter Axis 1. END B aktiv (low) ist. Gefunden am 16. November 2013.
- Grund: Dieser Schalter ist mit einem Boot-Setting-Pin des FPGA verbunden
- Problemlösung/Workaround: Verwenden Sie einen anderen Schalterpin, oder schließen Sie nur einen normalerweise offenen Schalter an diesen Schaltereingangspin an.

6.3. GS2 VFD-Treiber

Dies ist ein nicht-Echtzeit HAL-Programm für die GS2-Serie von VFDs bei Automation Direct. [1]

Diese Komponente wird mit dem halcmd-Befehl "loadusr" geladen:

```
loadusr -Wn spindle-vfd gs2_vfd -n spindle-vfd
```

Der obige Befehl lautet: loadusr, wait for named to load, component gs2_vfd, named spindle-vfd. Der HAL-Befehl "loadusr" ist im Kapitel loadusr beschrieben.

6.3.1. Kommandozeilen-Optionen

- *-b* oder *--bits* <*n*> (Voreinstellung: 8) Setzt die Anzahl der Datenbits auf *n*, wobei *n* von 5 bis einschließlich 8 reichen darf.
- *-d* oder *--device* <*path*> (Standard /dev/ttyS0) Legt den Dateipfad fest zum Ansprechen des seriellen Geräts.
- *-g* oder *--debug* Schaltet Debug-Meldungen ein. Dadurch wird auch das Verbose-Flag gesetzt. Der Debug-Modus bewirkt, dass alle Modbus-Meldungen in Hexadezimalschrift auf dem Terminal ausgegeben werden.
- *-n* oder *--name* <*string*> (Voreinstellung: gs2_vfd) Setzt den Namen des HAL-Moduls. Der HAL-Comp-Name wird auf <string> gesetzt, und alle Pin- und Parameternamen beginnen mit <string>.
- *-p* oder *--parity {even,odd,none}* (Voreinstellung: odd) Setzt die serielle Parität auf gerade (engl. even), ungerade (engl. odd) oder keine (engl. none).
- *-r* oder *--rate* <*n*> (Voreinstellung 38400) Setzt die Baudrate auf *n*. Es ist ein Fehler, wenn die Rate nicht eine der folgenden ist: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- *-s* oder *--stopbits {1,2}* (Voreinstellung: 1) Setzt die Anzahl von Stopbits auf 1 oder 2
- *-t* oder *--target* <*n*> (Voreinstellung: 1) Legt die MODBUS-Zielnummer (slave) fest. Diese muss mit der Gerätenummer übereinstimmen, die Sie am GS2 eingestellt haben.
- *-v* oder *--verbose* Schaltet Debug-Meldungen ein.
- *-A* oder *--accel-seconds* <*n*> (Voreinstellung: 10.0) Sekunden um die Spindel von 0 auf max. U/min (engl. RPM) zu beschleunigen.
- *-D* oder *--decel-seconds* <*n*> (Voreinstellung: 0.0) Sekunden, um die Spindel von max. U/min auf 0 abzubremsen. Bei einer Einstellung von 0.0 kann die Spindel ohne kontrollierte Abbremsung bis zum Stillstand ausrollen.
- *-R* oder *--braking-resistor* (engl. für Bremswiderstand) Dieses Argument sollte verwendet werden,

wenn ein Bremswiderstand am GS2-VFD installiert ist (siehe Anhang A des GS2-Handbuchs). Es deaktiviert die Überspannungsabschaltung bei Verzögerung (siehe GS2 Modbus Parameter 6.05), so dass der Frequenzumrichter auch in Situationen, in denen der Motor eine hohe Spannung zurückspeist, weiter bremsen kann. Die rückgespeiste Spannung wird sicher in den Bremswiderstand abgeleitet.

NOTE Bei seriellen Konfigurationsfehlern kann das Einschalten von verbose zu einer Flut von Timeout-Fehlern führen.

6.3.2. Pins

Dabei ist <*name*> "gs2_vfd" oder der Name, der beim Laden mit der Option `-n` angegeben wurde:

- <*name*>.*DC-bus-volts* (real, out) DC bus voltage of the VFD
- <*name*>.*at-speed* (bool, out) when drive is at commanded speed
- <*name*>.*err-reset* (bool, in) reset errors sent to VFD
- <*name*>.*firmware-revision* (sint, out) from the VFD
- <*name*>.*frequency-command* (real, out) from the VFD
- <*name*>.*frequency-out* (real, out) from the VFD
- <*name*>.*is-stopped* (bool, out) when the VFD reports 0 Hz output
- <*name*>.*load-percentage* (real, out) from the VFD
- <*name*>.*motor-RPM* (real, out) from the VFD
- <*name*>.*output-current* (real, out) from the VFD
- <*name*>.*output-voltage* (real, out) from the VFD
- <*name*>.*power-factor* (real, out) from the VFD
- <*name*>.*scale-frequency* (real, out) from the VFD
- <*name*>.*speed-command* (real, in) speed sent to VFD in RPM It is an error to send a speed faster than the Motor Max RPM as set in the VFD.
- <*name*>.*spindle-fwd* (bool, in) 1 for FWD and 0 for REV sent to VFD
- <*name*>.*spindle-rev* (bool, in) 1 for REV and 0 if off
- <*name*>.*spindle-on* (bool, in) 1 for ON and 0 for OFF sent to VFD
- <*name*>.*status-1* (sint, out) Drive Status of the VFD (see the GS2 manual)
- <*name*>.*status-2* (sint, out) Drive Status of the VFD (see the GS2 manual)

NOTE Der Statuswert ist die Summe aller Bits, die eingeschaltet sind. So ist eine 163, die bedeutet, dass sich das Laufwerk im Betriebsmodus befindet, die Summe aus 3 (Betrieb) + 32 (über die serielle Schnittstelle eingestellte Frequenz) + 128 (über die serielle Schnittstelle eingestellter Betrieb).

6.3.3. Parameter

Dabei ist <*name*> gs2_vfd oder der Name, der beim Laden mit der Option `-n` angegeben wurde:

- <*name*>.*error-count* (sint, RW)
- <*name*>.*loop-time* (real, RW) how often the modbus is polled (default: 0.1)
- <*name*>.*nameplate-HZ* (real, RW) Nameplate Hz of motor (default: 60)
- <*name*>.*nameplate-RPM* (real, RW) Nameplate RPM of motor (default: 1730)
- <*name*>.*retval* (sint, RW) the return value of an error in HAL
- <*name*>.*tolerance* (real, RW) speed tolerance (default: 0.01)
- <*name*>.*ack-delay* (sint, RW) number of read/write cycles before checking at-speed (default 2)

Ein Beispiel für die Verwendung dieser Komponente zum Antreiben einer Spindel finden Sie im Beispiel GS2 Spindel.

6.4. HAL Treiber für Raspberry Pi GPIO-Pins

Anmerkung: Dieser Treiber wird nicht in disk images (Dateien, die auf eine Festplatte/USB-stick 1:1 kopiert werden können, um damit den Rechner zu starten) zusammengestellt, die auf nicht-ARM CPUS abzielen. Es ist nur wirklich beabsichtigt, mit dem Raspberry Pi zu arbeiten. Sie kann (oder auch nicht) an ähnlichen boards oder direkten Klonen arbeiten.

6.4.1. Zweck

Dieser Treiber ermöglicht die Verwendung der Rapberry Pi GPIO Pins in einer Weise analog zum Parallelport-Treiber auf x86 PCs. Es kann die gleichen Schrittgeneratoren, Encoder-Zähler und ähnliche Komponenten verwenden.

6.4.2. Anwendung

```
loadrt hal_pi_gpio dir=0x13407 exclude=0x1F64BF8
```

Die "dir"-Maske (engl. dir mask) bestimmt, ob die Pins Eingänge oder Ausgänge sind, die Ausschluss-Maske (engl. exclude mask) verhindert, dass der Treiber die Pins verwendet (und ermöglicht so, dass sie für ihre normalen RPi-Zwecke wie SPI oder UART verwendet werden können).

Die Maske kann in Dezimal oder Hexadezimal (Hex kann einfacher sein da kein Übertrag von niederen zu höheren Ziffern stattfindet).

Um den Wert der Maskierungeen zu ermitteln, summieren Sie die (Hexa-)Dezimalwerte für alle Pins, die als Ausgabe konfiguriert werden sollen, und analog für alle Pins, die gemäß der folgenden Tabelle ausgeschlossen werden sollten.

Table 40. GPIO Maskierungen - Zuordnung (engl. mapping) von GPIO-Nummern (linkste Spalte) zu physikalischen Pin-Nummern, wie auf dem Raspberry Pi-Board (rechtsste Spalte) gedruckt und die Dezimal-

/Hexadezimalwerte, die zum Wert der Maske beitragen.

GPIO Nummer	Dezimal	Hexadezimal	Pin-Nummer
2	1	0x00000001	3
3	2	0x00000002	5
4	4	0x00000004	7
5	8	0x00000008	29
6	16	0x00000010	31
7	32	0x00000020	26
8	64	0x00000040	24
9	128	0x00000080	21
10	256	0x00000100	19
11	512	0x00000200	23
12	1024	0x00000400	32
13	2048	0x00000800	33
14	4096	0x00001000	8
15	8192	0x00002000	10
16	16384	0x00004000	36
17	32768	0x00008000	11
18	65536	0x00010000	12
19	131072	0x00020000	35
20	262144	0x00040000	38.
21	524288	0x00080000	40
22	1048576	0x00100000	15
23	2097152	0x00200000	16
24	4194304	0x00400000	18
25	8388608	0x00800000	22
26	16777216	0x01000000	37
27	33554432	0x02000000	13

Note: Bei der Berechnung der Masken werden GPIO-Nummern verwendet, der individuelle Maskenwert eines Pins ergibt sich hierbei als 2^(*GPIO number* - 2), während bei der Benennung der HAL Pins es die Raspberry Pi Header Pin Zahlen sind.

So, wenn Sie zum Beispiel GPIO 17 als Ausgang aktivieren (dir=0x8000), dann wird dieser Ausgang von dem HAL Pin **hal_pi_gpio.pin-11-out** gesteuert.

6.4.3. Pins

- hal_pi_gpio.pin-*NN*-out
- hal_pi_gpio.pin-*NN*-in

Abhängig von den "dir" und Ausschluss (engl. exclude) Maskierungen.

The standard timing pins which are created for all components also exist:

- hal_pi_gpio.read.time
- hal_pi_gpio.read.tmax
- hal_pi_gpio.read.tmax-increased
- hal_pi_gpio.write.time
- hal_pi_gpio.write.tmax
- hal_pi_gpio.write.tmax-increased

6.4.4. Parameter

Keinen.

6.4.5. Funktionen

- `hal_pi_gpio.read` - Fügen Sie dies dem *base thread* hinzu, um die HAL-Pin-Werte zu aktualisieren und physikalische Eingangswerte anzupassen.
- `hal_pi_gpio.write` - Fügen Sie dies dem *base thread* hinzu, um die physikalischen Pins zu aktualisieren indem sie sich den HAL-Werten anpassen.

Typischerweise wird die *read* Funktion früh in der Callliste (engl. call list) stehen, vor irgendwelchen Encoder-Zählern, und die *write* Funktion nachfolgend in der call list, wohl auch nach *stepgen.make-pulses*.

6.4.6. Pin-Nummerierung

Der GPIO-Steckverbinder und der Pinout sind seit rund 2015 konsistent. Diese ältere Pi-Modelle sind wahrscheinlich ohnehin eine schlechte Wahl für LinuxCNC. Dieser Treiber ist jedoch dazu ausgelegt, mit diesen zu arbeiten, wird die beiden alternativen Pinouts entsprechend erkennen und korrekt konfigurieren.

Die aktuelle Pinout-Mapping zwischen GPIO-Nummern und Pin-Nummern ist in der obigen Tabelle beschrieben.

Beachten Sie, dass die config string GPIO-Nummern verwendet, aber sobald der Treiber geladen ist,

beziehen sich die HAL-Pin-Namen auf Stecker-Pin-Nummern.

Dies kann logischer sein, als es zunächst erscheint. Bei der Einrichtung müssen Sie genug Pins jeden Typs konfigurieren, während Sie vermeiden wollen, andere Funktionen zu überschreiben, die Ihr System benötigt. Dann, wenn der Treiber geladen wird, in der HAL-Schicht, wollen Sie nur wissen, womit Sie die Drähte für jeden HAL-Pin verbinden.

6.4.7. Bekannte Probleme

Derzeit (2023-07-16) scheint dieser Treiber nur auf Raspbian zu arbeiten, da das generische Debian-Image die richtigen Schnittstellen in /dev/gpiomem nicht einrichtet und den Zugriff auf die /sys/mem-Schnittstelle einschränkt.

6.5. Generische Treiber für alle GPIO unterstützt von gpiod.

Dieser Treiber wurde auf dem Raspberry Pi getestet und sollte auch an Banana Pi, BeagleBone, Pine64 (u.a.) und andere single board computer und potenziell auf anderen Plattformen arbeiten.

6.5.1. Zweck

Dieser Treiber ermöglicht die Verwendung von GPIO-Pins irgendwie ähnlich zum Parallelport-Treiber auf x86 PCs. Es kann die gleichen Schrittgeneratoren, Encoderzähler und ähnliche Komponenten verwenden.

6.5.2. Anwendung

```
loadrt hal_gpio inputs=GPIO5,GPIO6,GPIO12,GPIO13,GPIO16,GPIO17,GPIO18,GPIO19 \
                outputs=GPIO20,GPIO21,GPIO22,GPIO23,GPIO24,GPIO25,GPIO26,GPIO27 \
                invert=GPIO20,GPIO27 \
                reset=GPIO21,GPIO22
```

Dieser Treiber basiert auf der libgpiod-dev-Bibliothek und dem gpiod-Paket, das eine Reihe von Diensten zur Konfiguration und Abfrage von GPIO enthält. Die GPIO-Pin-Namen in der oben angegebenen Zeile "loadrt" des HAL sollten die Namen des `gpioinfo`-Befehls sein.

Beispiel Ausgabe (gekürzt):

```
$ gpioinfo
gpiochip0 - 54 lines:
    line   0:      "ID_SDA"       unused   input  active-high
    line   1:      "ID_SCL"       unused   input  active-high
    line   2:        "SDA1"       unused   input  active-high
    line   3:        "SCL1"       unused   input  active-high
    line   4:  "GPIO_GCLK"       unused   input  active-high
    line   5:       "GPIO5"       unused   input  active-high
    line   6:       "GPIO6"       unused   input  active-high
    line   7:  "SPI_CE1_N"       unused   input  active-high
    line   8:  "SPI_CE0_N"       unused   input  active-high
    line   9:   "SPI_MISO"       unused   input  active-high
```

```
    line  10:   "SPI_MOSI"       unused   input  active-high
    line  11:   "SPI_SCLK"       unused   input  active-high
    line  12:     "GPIO12"       unused   input  active-high
    line  13:     "GPIO13"       unused   input  active-high
    line  14:       "TXD1"       unused   input  active-high
    line  15:       "RXD1"       unused   input  active-high
    line  16:     "GPIO16"       unused   input  active-high
    line  17:     "GPIO17"       unused   input  active-high
    line  18:     "GPIO18"       unused   input  active-high
    line  19:     "GPIO19"       unused   input  active-high
    line  20:     "GPIO20"       unused  output  active-high
...
```

Eine Liste von Eingangs- und/oder Ausgangspins ist wie in der obigen Probe dargestellt anzugeben. Das \-Zeichen wird für die Linienfortführung in HAL verwendet und dient zur Verbesserung der Lesbarkeit. Die Pin-Namen sind case-sensitive und es darf keine Leerzeichen in den Strings geben, auch nicht zwischen den Komma-getrennten Pin-Listen der "=" Zeichen.

Weitere Modifikatoren sind

invert

(nur für Ausgänge gültig). Invertiert den Sinn (engl. sense) des physikalischen Pins relativ zum Wert in HAL.

reset

(nur für Ausgänge gültig). Werden der Liste "reset" beliebige Pins zugeordnet als ein HAL-Parameter **hal_gpio.reset_ns** wird erstellt. Dies hat keinen Effekt, wenn die Funktion **hal_gpio.reset** einem Echtzeit-Gewinde hinzugefügt wird. Dies sollte nach der Funktion **hal_gpio.write** platziert werden und muss im gleichen thread ausgeführt werden. Das Verhalten dieser Funktion entspricht der gleichen Funktion im **hal_parport**-Treiber und ermöglicht einen Schrittimpuls jeden Thread-Zyklus. Wird die **hal_gpio.reset_ns**-Zeit länger als 1/4 der Periodendauer des Threads eingestellt, dem sie hinzugefügt wird, so wird der Wert auf 1/4 der Threadperiode reduziert. Es gibt eine untere Grenze, wie lange der Puls sein kann. Bei 8 Pins in der Ausgabeliste kann die Pulsbreite beispielsweise auf einem RPi4 nicht weniger als 5000 ns reduzieren.

Die folgenden Funktionen werden in allen Versionen akzeptiert, sind jedoch nur wirksam, wenn eine Version von libgpiod_dev >= 1.6 installiert ist. Sie sollten in gleicher Weise wie die oben beschriebenen Parameter verwendet werden und die elektrischen Parameter der GPIO-Pins ändern, **wenn** dies durch die Hardware unterstützt wird.

opendrain

opensource

biasdisable

pulldown

pullup

Die Version von libgpiod-dev installiert kann durch den Befehl `gpioinfo -v` bestimmt werden

6.5.3. Pins

- hal_gpio.NAME-in - HAL_OUT Der Wert eines Eingangspins, der in HAL dargestellt ist
- hal_gpio.NAME-in-not - HAL_OUT Eine invertierte Version von oben, für Komfort
- hal_gpio.NAME-out - HAL_IN use this pin to transfer a HAL bool value to a physical output

6.5.4. Parameter

- hal_gpio.reset_ns - HAL_RW - "setp" diesen Parameter, um die Pulslänge von Pins zu steuern, die der "Reset"-Liste hinzugefügt wurden. Der Wert wird zwischen 0 und Thread-Periode / 4 begrenzt.

6.5.5. Funktionen

- *hal_gpio_gpio.read* - Fügen Sie dies dem *base thread* hinzu, um die HAL-Pin-Werte zu aktualisieren und physikalische Eingangswerte anzupassen.
- *hal_pi_gpio.write* - Fügen Sie dies dem *base thread* hinzu, um die physikalischen Pins zu aktualisieren indem sie sich den HAL-Werten anpassen.
- *hal_gpio.reset* - Nur exportiert, wenn in der Reset-Liste Pins definiert sind. Dies sollte nach der Funktion "write" platziert werden und sollte im gleichen thread sein.

Typischerweise wird die *read* Funktion früh in der Callliste (engl. call list) stehen, vor irgendwelchen Encoder-Zählern, und die *write* Funktion nachfolgend in der call list, wohl auch nach `stepgen.make-pulses`.

6.5.6. Pin Identifizierung

Verwenden Sie die Pin-Namen, die vom Dienstprogramm `gpioinfo` zurückgegeben wurden. Dies verwendet die Daten aus dem Geräteverzeichnis-Baum (engl. device tree). Wenn das installierte Betriebssystem keine Gerätebaum-Datenbank hat, werden die Pins alle als "unnamed" (oder ähnlich) bezeichnet und dieser Treiber kann nicht verwendet werden.

Ein weiteres Update dieses Treibers könnte den Zugriff durch Indexnummer ermöglichen, aber dies wird derzeit nicht unterstützt.

6.5.7. Fehlerfindung bei Problemen mit Zugriffsberechtigungen.

Wenn beim Laden des Treibers Zugriffsfehler (engl. "access denied") zurückgegeben werden, versuchen Sie folgendes Rezept: (Sollte für Raspbian nicht benötigt werden und ist für aktuelle GPIO-Chipnamen auf Nicht-Pi-Plattformen anzupassen)

1. Erstellen Sie eine neue Gruppe `gpio` mit dem Befehl

```
sudo groupadd gpio
```

2. Dann, um Berechtigungen für die "gpio"-Gruppe einzurichten, erstellen Sie eine Datei namens `90-gpio-access` im `/etc/udev/rules.d/`-Verzeichnis mit den folgenden Inhalten (dies wird von der Raspbian-Installation kopiert)

```
SUBSYSTEM=="bcm2835-gpiomem", GROUP="gpio", MODE="0660"
SUBSYSTEM=="gpio", GROUP="gpio", MODE="0660"
SUBSYSTEM=="gpio*", PROGRAM="/bin/sh -c '\
    chown -R root:gpio /sys/class/gpio && chmod -R 770 /sys/class/gpio;\
    chown -R root:gpio /sys/devices/virtual/gpio &&\
    chmod -R 770 /sys/devices/virtual/gpio;\
    chown -R root:gpio /sys$devpath && chmod -R 770 /sys$devpath\
'"

SUBSYSTEM=="pwm*", PROGRAM="/bin/sh -c '\
    chown -R root:gpio /sys/class/pwm && chmod -R 770 /sys/class/pwm;\
    chown -R root:gpio /sys/devices/platform/soc/*.pwm/pwm/pwmchip* &&\
    chmod -R 770 /sys/devices/platform/soc/*.pwm/pwm/pwmchip*\
'"
```

3. Fügen Sie den UNIX Benutzer, der LinuxCNC anwendet, der UNIX Gruppe `gpio` hinzu mit

```
sudo usermod -aG gpio <username>
```

6.5.8. Autor

Andy Pugh

6.5.9. Bekannte Probleme

Derzeit keine.

6.6. Mesa HostMot2-Treiber

6.6.1. Einführung

HostMot2 ist eine FPGA-Konfiguration, die von Mesa Electronics für ihre *Anything I/O* -Bewegungssteuerungskarten entwickelt wurde. Die Firmware ist quelloffen, portabel und flexibel. Sie kann (zur Kompilierzeit) mit null oder mehr Instanzen (ein zur Laufzeit erstelltes Objekt) von jedem der verschiedenen Module konfiguriert werden: Encoder (Quadraturzähler), PWM-Generatoren und Schritt-/Differenzgeneratoren. Die Firmware kann (zur Laufzeit) so konfiguriert werden, dass jede dieser Instanzen mit Pins an den E/A-Headern verbunden wird. Die nicht von einer Modulinstanz angesteuerten E/A-Pins werden zu allgemeinen bidirektionalen digitalen E/A Pins.

6.6.2. Firmware-Binärdateien

50 Pin Header FPGA-Karten

Mehrere vorkompilierte HostMot2-Firmware-Binärdateien sind für die verschiedenen Anything-I/O-

Karten verfügbar. Diese Liste ist unvollständig, schauen Sie in der hostmot2-firmware-Distribution nach aktuellen Firmware-Listen.

- 3x20 (144 E/A-Pins): mit hm2_pci-Modul
  - 24-Kanal-Servo
  - 16-Kanal-Servo und 24 Schritt/Richtung (engl. step/dir)-Generatoren
- 5I22 (96 E/A-Stifte): mit hm2_pci-Modul
  - 16-Kanal-Servo
  - 8-Kanal-Servo plus 24 Step/Dir-Generatoren
- 5I20, 5I23, 4I65, 4I68 (72 E/A-Pins): mit hm2_pci-Modul
  - 12-Kanal-Servo
  - 8-Kanal-Servo plus 4 Step/Dir-Generatoren
  - 4-Kanal-Servo plus 8 Step/Dir-Generatoren
- 7I43 (48 E/A-Pins): Verwendung des Moduls hm2_7i43
  - 8-Kanal-Servo (8 PWM-Generatoren und 8 Encoder)
  - 4-Kanal-Servo plus 4 Step/Dir-Generatoren

DB25 FPGA-Karten

Die 5I25 Superport FPGA-Karte ist beim Kauf bereits vorprogrammiert und benötigt keine binäre Firmware.

6.6.3. Installieren der Firmware

Je nachdem, wie Sie LinuxCNC installiert haben, müssen Sie möglicherweise den Synaptic Package Manager aus dem Systemmenü öffnen und das Paket für Ihre Mesa-Karte installieren. Der schnellste Weg, um sie zu finden, ist eine Suche nach "hostmot2" in der Synaptic Package Manager zu tun. Markieren Sie die Firmware für die Installation, und wenden Sie sie an.

6.6.4. Laden von HostMot2

Die LinuxCNC-Unterstützung für die HostMot2-Firmware ist in einen generischen Treiber namens *hostmot2* und zwei Low-Level-I/O-Treiber für die Anything-I/O-Karten aufgeteilt. Die Low-Level-I/O-Treiber sind *hm2_7i43* und *hm2_pci* (für alle PCI- und PC-104/Plus-basierten Anything-I/O-Karten). Der hostmot2-Treiber muss zuerst mit einem HAL-Befehl wie diesem geladen werden:

```
loadrt hostmot2
```

Siehe die Manpage zu hostmot2(9) für Details.

Der Hostmot2-Treiber für sich allein tut nichts, er braucht Zugang zu den tatsächlichen Boards, auf denen die HostMot2-Firmware läuft. Die Low-Level-I/O-Treiber stellen diesen Zugang zur Verfügung. Die Low-Level-I/O-Treiber werden mit Befehlen wie diesem geladen:

```
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT
       num_encoders=3 num_pwmgens=3 num_stepgens=1"
```

Die Konfigurationsparameter sind in der Manpage zu hostmot2 beschrieben.

6.6.5. Watchdog

Die HostMot2-Firmware kann ein Watchdog-Modul enthalten; wenn dies der Fall ist, wird es vom Hostmot2-Treiber verwendet.

Der Watchdog muss von Zeit zu Zeit von LinuxCNC gestreichelt werden, sonst beißt er. Die hm2 Schreibfunktion (siehe unten) streichelt den Watchdog.

Wenn der Watchdog anspricht, werden alle E/A-Pins des Boards von ihren Modulinstanzen getrennt und werden zu hochohmigen Eingängen (hochgezogen). Der Zustand der HostMot2-Firmware-Module wird nicht gestört (mit Ausnahme der Konfiguration der I/O-Pins). Die Encoder-Instanzen zählen weiterhin die Quadraturimpulse, und die Pwm- und Schrittgeneratoren erzeugen weiterhin Signale (die nicht an die Motoren weitergeleitet werden, da die I/O-Pins zu Eingängen geworden sind).

Durch das Zurücksetzen des Watchdogs werden die E/A-Pins auf die zum Zeitpunkt des Ladens gewählte Konfiguration zurückgesetzt.

Wenn die Firmware einen Watchdog enthält, werden die folgenden HAL-Objekte exportiert:

Pins

- *has_bit* - (bool i/o) True if the watchdog has bit, False if the watchdog has not bit. If the watchdog has bit and the has_bit bool is True, the user can reset it to False to resume operation.

Parameter

- *timeout_ns* - (uint read/write) Watchdog timeout, in nanoseconds. This is initialized to 5,000,000 (5 milliseconds) at module load time. If more than this amount of time passes between calls to the hm2 write function, the watchdog will bite.

6.6.6. HostMot2-Funktionen

- *hm2_<BoardType>.<BoardNum>.read* - Lesen aller Eingänge, Aktualisieren der Eingangs-HAL-Pins.
- *hm2_<BoardType>.<BoardNum>.write* - Alle Ausgänge schreiben.
- *hm2_<BoardType>.<BoardNum>.read_gpio* - Liest nur die GPIO-Eingangsstifte. (Diese Funktion ist auf der 7I43 aufgrund der Einschränkungen des EPP-Busses nicht verfügbar.)
- *hm2_<BoardType>.<BoardNum>.write_gpio* - Schreibt nur die GPIO-Steuerregister und Ausgangspins. (Diese Funktion ist auf der 7I43 aufgrund der Einschränkungen des EPP-Busses nicht verfügbar.)

NOTE Die obigen Funktionen *read_gpio* und *write_gpio* sollten normalerweise nicht benötigt werden, da die GPIO-Bits zusammen mit allem anderen in den obigen

> Standardfunktionen *read* und *write* gelesen und geschrieben werden, die normalerweise im Servo-Thread ausgeführt werden.
>
> Die Funktionen *read_gpio* und *write_gpio* wurden für den Fall bereitgestellt, dass eine sehr schnelle (häufig aktualisierte) E/A benötigt wird. Diese Funktionen sollten im Basis-Thread ausgeführt werden. Wenn Sie dies benötigen, senden Sie uns bitte eine E-Mail und teilen Sie uns mit, um welche Anwendung es sich handelt.

6.6.7. Pinbelegungen

Der hostmot2-Treiber hat keine bestimmte Pinbelegung. Die Pinbelegung ergibt sich aus der Firmware, die der hostmot2-Treiber an die Anything I/O-Karte sendet. Jede Firmware hat eine andere Pinbelegung, und die Pinbelegung hängt davon ab, wie viele der verfügbaren Encoder, pwmgens und stepgens verwendet werden. Um eine Pinout-Liste für Ihre Konfiguration nach dem Laden von LinuxCNC im Terminalfenster zu erhalten, geben Sie ein:

```
dmesg > hm2.txt
```

Die resultierende Textdatei enthält viele Informationen sowie die Pinbelegung für den HostMot2 und alle Fehler- und Warnmeldungen.

Um das Durcheinander zu reduzieren, indem der Nachrichtenpuffer vor dem Laden von LinuxCNC gelöscht wird, geben Sie Folgendes in das Terminalfenster ein:

```
sudo dmesg -c
```

Nun, wenn Sie LinuxCNC ausführen, erhalten SIe über *dmesg > hm2.txt* im Terminal nur die Informationen seit der Zeit, die LinuxCNC läuft zusammen mit Ihrem Pinout. Die Datei wird im aktuellen Verzeichnis des Terminalfensters liegen. Jede Zeile enthält den Kartennamen, die Kartennummer, die E/A-Pin-Nummer, den Stecker und den Pin sowie die Verwendung. Anhand dieses Ausdrucks können Sie die physischen Verbindungen zu Ihrer Karte entsprechend Ihrer Konfiguration erkennen.

Ein Beispiel für eine 5I20-Konfiguration:

```
[HOSTMOT2]
DRIVER=hm2_pci
BOARD=5i20
CONFIG="firmware=hm2/5i20/SVST8_4.BIT num_encoders=1 num_pwmgens=1 num_stepgens=3"
```

Die obige Konfiguration ergab diesen Ausdruck.

```
[ 1141.053386] hm2/hm2_5i20.0: 72 I/O Pins used:
[ 1141.053394] hm2/hm2_5i20.0: IO Pin 000 (P2-01): IOPort
[ 1141.053397] hm2/hm2_5i20.0: IO Pin 001 (P2-03): IOPort
[ 1141.053401] hm2/hm2_5i20.0: IO Pin 002 (P2-05): Encoder #0, pin B (Input)
[ 1141.053405] hm2/hm2_5i20.0: IO Pin 003 (P2-07): Encoder #0, pin A (Input)
[ 1141.053408] hm2/hm2_5i20.0: IO Pin 004 (P2-09): IOPort
```

```
[ 1141.053411] hm2/hm2_5i20.0: IO Pin 005 (P2-11): Encoder #0, pin Index (Input)
[ 1141.053415] hm2/hm2_5i20.0: IO Pin 006 (P2-13): IOPort
[ 1141.053418] hm2/hm2_5i20.0: IO Pin 007 (P2-15): PWMGen #0, pin Out0 (PWM or Up)
(Output)
[ 1141.053422] hm2/hm2_5i20.0: IO Pin 008 (P2-17): IOPort
[ 1141.053425] hm2/hm2_5i20.0: IO Pin 009 (P2-19): PWMGen #0, pin Out1 (Dir or Down)
(Output)
[ 1141.053429] hm2/hm2_5i20.0: IO Pin 010 (P2-21): IOPort
[ 1141.053432] hm2/hm2_5i20.0: IO Pin 011 (P2-23): PWMGen #0, pin Not-Enable (Output)
<snip>...
[ 1141.053589] hm2/hm2_5i20.0: IO Pin 060 (P4-25): StepGen #2, pin Step (Output)
[ 1141.053593] hm2/hm2_5i20.0: IO Pin 061 (P4-27): StepGen #2, pin Direction (Output)
[ 1141.053597] hm2/hm2_5i20.0: IO Pin 062 (P4-29): StepGen #2, pin (unused) (Output)
[ 1141.053601] hm2/hm2_5i20.0: IO Pin 063 (P4-31): StepGen #2, pin (unused) (Output)
[ 1141.053605] hm2/hm2_5i20.0: IO Pin 064 (P4-33): StepGen #2, pin (unused) (Output)
[ 1141.053609] hm2/hm2_5i20.0: IO Pin 065 (P4-35): StepGen #2, pin (unused) (Output)
[ 1141.053613] hm2/hm2_5i20.0: IO Pin 066 (P4-37): IOPort
[ 1141.053616] hm2/hm2_5i20.0: IO Pin 067 (P4-39): IOPort
[ 1141.053619] hm2/hm2_5i20.0: IO Pin 068 (P4-41): IOPort
[ 1141.053621] hm2/hm2_5i20.0: IO Pin 069 (P4-43): IOPort
[ 1141.053624] hm2/hm2_5i20.0: IO Pin 070 (P4-45): IOPort
[ 1141.053627] hm2/hm2_5i20.0: IO Pin 071 (P4-47): IOPort
[ 1141.053811] hm2/hm2_5i20.0: registered
[ 1141.053815] hm2_5i20.0: initialized AnyIO board at 0000:02:02.0
```

NOTE Der I/O Pin nnn entspricht der Pin-Nummer, die auf dem HAL Configuration Bildschirm für GPIOs angezeigt wird. Einige der StepGen, Encoder und PWMGen werden auch als GPIOs im HAL-Konfigurationsbildschirm angezeigt.

6.6.8. PIN-Dateien

Die Standard-Pinbelegung ist in einer .PIN-Datei (menschenlesbarer Text) beschrieben. Wenn Sie ein Firmware-Paket installieren, werden die .PIN-Dateien in

```
/usr/share/doc/hostmot2-firmware-<board>/
```

6.6.9. Firmware

Die ausgewählte Firmware (.BIT-Datei) und Konfiguration wird beim Start von LinuxCNC von der PC-Hauptplatine auf die Mesa-Hauptplatine hochgeladen. Wenn Sie Run In Place verwenden, müssen Sie noch ein hostmot2-firmware-<board> Paket installieren. Weitere Informationen über Firmware und Konfiguration finden Sie im Abschnitt *Konfigurationen*.

6.6.10. HAL-Pins

Die HAL-Pins für jede Konfiguration können durch Öffnen von *Show HAL Configuration* aus dem Menü Maschine angezeigt werden. Alle HAL-Pins und Parameter sind dort zu finden. Die folgende Abbildung zeigt die oben verwendete 5I20-Konfiguration.

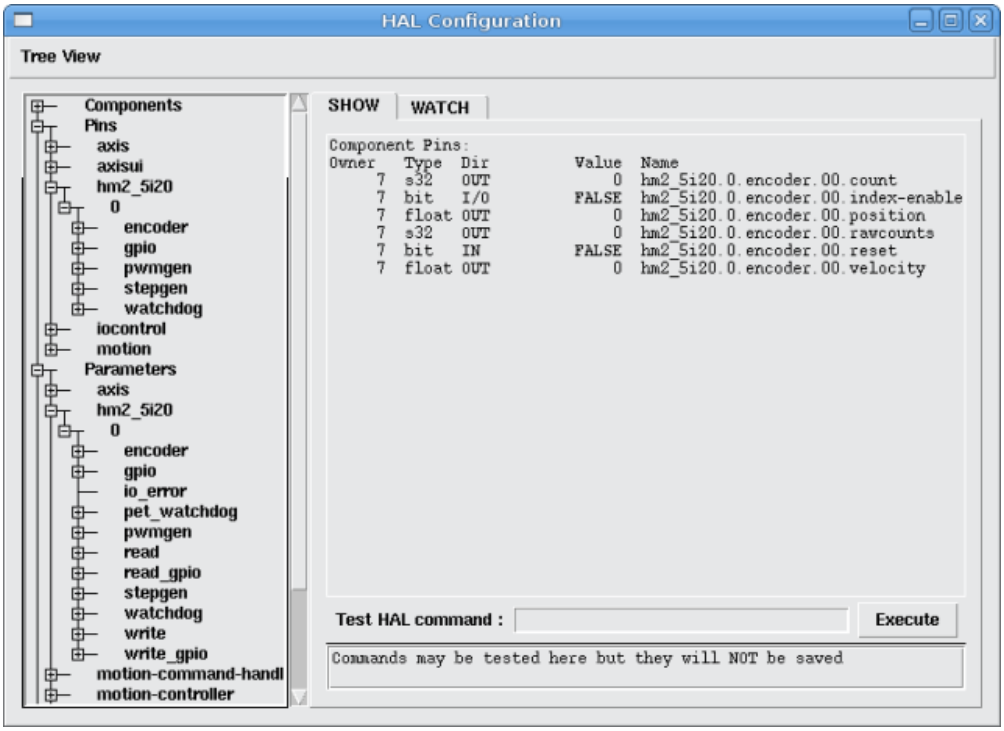


Figure 109. 5i20 HAL-Pins

6.6.11. Konfigurationen

Die Hostmot2-Firmware gibt es in verschiedenen Versionen, je nachdem, was Sie erreichen wollen. Sie können sich anhand des Namens einen Überblick verschaffen, wofür eine bestimmte Firmware geeignet ist. Schauen wir uns ein paar Beispiele an.

In der 7I43 (zwei Ports) wäre SV8 (*Servo 8*) für 8 Servos oder weniger, unter Verwendung des *klassischen* 7I33 4-Achsen-Servoboards (pro Port). 8 Servos würden also alle 48 Signale an den beiden Ports belegen. Wenn Sie aber nur 3 Servos benötigen, könnten Sie *num_encoders=3* und *num_pwmgens=3* sagen und 5 Servos mit je 6 Signalen wiederherstellen, wodurch Sie 30 Bits GPIO gewinnen.

Oder beim 5I22 (vier Anschlüsse) wäre SVST8_24 (*Servo 8, Stepper 24*) für 8 Servos oder weniger (wieder 7I33 x2) und 24 Stepper oder weniger (7I47 x2). Damit wären alle vier Ports belegt. Wenn man nur 4 Servos braucht, könnte man *num_encoders=4* und *num_pwmgens=4* sagen und 1 Port zurückgewinnen (und einen 7I33 sparen). Und wenn man nur 12 Stepper bräuchte, könnte man sagen *num_stepgens=12* und einen Port freigeben (und einen 7I47 sparen). Auf diese Weise können wir also zwei Ports (48 Bits) für GPIO einsparen.

Hier sind Tabellen mit den in den offiziellen Paketen verfügbaren Firmwares. Es kann zusätzliche Firmwares auf der Mesanet.com Website, die noch nicht in die LinuxCNC offiziellen Firmware-Pakete geschafft haben, daher schauen Sie auch dort nach.

3x20 (verschiedene 6 Anschlüsse) Standardkonfigurationen (3x20 ist in den Versionen mit 1M, 1,5M und 2M Gatter erhältlich. Bislang ist die gesamte Firmware in allen Gate-Größen verfügbar.)

Firmware	Encoder	PWMGen	StepGen	GPIO
SV24	24	24	0	0

Firmware	Encoder	PWMGen	StepGen	GPIO
SVST16_24	16	16	24	0

5I22 (4-Port PCI) Standardkonfigurationen (Die 5I22 ist in Versionen mit 1M und 1,5M Gatter erhältlich. Bislang ist die gesamte Firmware für alle Gate-Größen verfügbar.)

Firmware	Encoder	PWM	StepGen	GPIO
SV16	16	16	0	0
SVST2_4_7I47	4	2	4	72
SVST8_8	8	8	8	0
SVST8_24	8	8	24	0

5I23 (3-Port PCI) Standardkonfigurationen (Die 5I23 hat 400k Gatter.)

Firmware	Encoder	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST2_8	2	2	8 (tbl5)	12
SVST2_4_7I47	4	2	4	48
SV12_2X7I48_72	12	12	0	24
SV12IM_2X7I48_72	12 (+IM)	12	0	12
SVST4_8	4	4	8 (tbl5)	0
SVST8_4	8	8	4 (tbl5)	0
SVST8_4IM2	8 (+IM)	8	4	8
SVST8_8IM2	8 (+IM)	8	8	0
SVTP6_7I39	6	0 (6 BLDC)	0	0

5I20 (3-Port PCI) Standardkonfigurationen (Die 5I20 hat 200k Gatter.)

Firmware	Encoder	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST2_8	2	2	8 (tbl5)	12
SVST2_4_7I47	4	2	4	48
SV12_2X7I48_72	12	12	0	24
SV12IM_2X7I48_72	12 (+IM)	12	0	12

Firmware	Encoder	PWM	StepGen	GPIO
SVST8_4	8	8	4 (tbl5)	0
SVST8_4IM2	8 (+IM)	8	4	8

4I68 (3-Port PC/104) Standardkonfigurationen (Die 4I68 hat 400k Gatter.)

Firmware	Encoder	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST2_4_7I47	4	2	4	48
SVST4_8	4	4	8	0
SVST8_4	8	8	4	0
SVST8_4IM2	8 (+IM)	8	4	8
SVST8_8IM2	8 (+IM)	8	8	0

4i65 (3-Port PC/104) Standardkonfigurationen (Die 4i65 hat 200k Gates.)

Firmware	Encoder	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST8_4	8	8	4	0
SVST8_4IM2	8 (+IM)	8	4	8

7I43 (2 Anschlüsse parallel) 400k-Gate-Versionen, Standardkonfigurationen

Firmware	Encoder	PWM	StepGen	GPIO
SV8	8	8	0	0
SVST4_4	4	4	4 (tbl5)	0
SVST4_6	4	4	6 (tbl3)	0
SVST4_12	4	4	12	0
SVST2_4_7I47	4	2	4	24

7I43 (2 Anschlüsse parallel) 200k-Gate-Versionen, Standardkonfigurationen

Firmware	Encoder	PWM	StepGen	GPIO
SV8	8	8	0	0
SVST4_4	4	4	4 (tbl5)	0
SVST4_6	4	4	6 (tbl3)	0

Firmware	Encoder	PWM	StepGen	GPIO
SVST2_4_7I47	4	2	4	24

Auch wenn mehrere Karten die gleiche .BIT-Datei haben, können Sie keine .BIT-Datei verwenden, die nicht für diese Karte bestimmt ist. Verschiedene Karten haben unterschiedliche Taktfrequenzen, also stellen Sie sicher, dass Sie die richtige .BIT-Datei für Ihre Karte laden. Benutzerdefinierte hm2-Firmwares können für spezielle Anwendungen erstellt werden und Sie können einige benutzerdefinierte hm2-Firmwares in den Verzeichnissen mit den Standard-Firmwares sehen.

Wenn Sie den Board-Treiber (hm2_pci oder hm2_7i43) laden, können Sie ihn anweisen, die Instanzen der drei primären Module (pwmgen, stepgen und encoder) zu deaktivieren, indem Sie die Anzahl niedriger setzen. Alle E/A-Pins, die zu deaktivierten Modulinstanzen gehören, werden zu GPIOs.

6.6.12. GPIO

Allgemeine E/A-Pins auf der Karte, wenn nicht von einer Modulinstanz verwendet, werden als *volle* GPIO-Pins an HAL exportiert. Full-GPIO-Pins können zur Laufzeit als Eingänge, Ausgänge oder Open Drains konfiguriert werden und verfügen über eine HAL-Schnittstelle, die diese Flexibilität offenlegt. E/A-Pins einer aktiven Modulinstanz sind durch die Anforderungen dieses sie besitzenden Moduls eingeschränkt und haben eine eingeschränkte HAL-Schnittstelle.

GPIOs haben Namen wie *hm2_<BoardType>.<BoardNum>.gpio.<IONum>*. IONum ist eine dreistellige Zahl. Die Zuordnung von IONum zu Stecker und Pin-auf-dem-Stecker wird in das Syslog geschrieben, wenn der Treiber geladen wird, und sie ist im Mesa-Handbuch für die Anything I/O-Boards dokumentiert.

Die hm2-GPIO-Darstellung ist den digitalen Eingängen und digitalen Ausgängen nachempfunden, die in der kanonischen Geräteschnittstelle (Teil des Dokuments HAL General Reference) beschrieben sind.

GPIO-Pins sind standardmäßig auf Eingang eingestellt.

Pins

- *in* - (bool, out) Normal state of the hardware input pin. Both full GPIO pins and I/O pins used as inputs by active module instances have this pin.
- *in_not* - (bool, out) Inverted state of the hardware input pin. Both full GPIO pins and I/O pins used as inputs by active module instances have this pin.
- *out* - (bool, in) Value to be written (possibly inverted) to the hardware output pin. Only full GPIO pins have this pin.

Parameter

- *invert_output* - (bool, rw) This parameter only has an effect if the *is_output* parameter is true. If this parameter is true, the output value of the GPIO will be the inverse of the value on the *out* HAL pin. Only full GPIO pins and I/O pins used as outputs by active module instances have this parameter. To invert an active module pin you have to invert the GPIO pin not the module pin.

- *is_opendrain* - (bool, rw) This parameter only has an effect if the *is_output* parameter is true. If this parameter is false, the GPIO behaves as a normal output pin: the I/O pin on the connector is driven to the value specified by the *out* HAL pin (possibly inverted), and the value of the *in* and *in_not* HAL pins is undefined. If this parameter is true, the GPIO behaves as an open-drain pin. Writing 0 to the *out* HAL pin drives the I/O pin low, writing 1 to the *out* HAL pin puts the I/O pin in a high-impedance state. In this high-impedance state the I/O pin floats (weakly pulled high), and other devices can drive the value; the resulting value on the I/O pin is available on the *in* and *in_not* pins. Only full GPIO pins and I/O pins used as outputs by active module instances have this parameter.
- *is_output* - (bool, rw) If set to 0, the GPIO is an input. The I/O pin is put in a high-impedance state (weakly pulled high), to be driven by other devices. The logic value on the I/O pin is available in the *in* and *in_not* HAL pins. Writes to the *out* HAL pin have no effect. If this parameter is set to 1, the GPIO is an output; its behavior then depends on the *is_opendrain* parameter. Only full GPIO pins have this parameter.

6.6.13. StepGen

StepGens haben Namen wie *hm2_<BoardType>.<BoardNum>.stepgen.<Instance>*. *Instance* ist eine zweistellige Nummer, die der HostMot2 stepgen-Instanznummer entspricht. Es gibt *num_stepgens* Instanzen, beginnend mit 00.

Jedes Stepgen belegt 2-6 E/A-Pins (die bei der Kompilierung der Firmware ausgewählt werden), verwendet aber derzeit nur zwei: Schritt- und Richtungsausgänge. Fußnote:[Derzeit unterstützt die Firmware mehrphasige Stepperausgänge, aber der Treiber nicht'. Interessierte Freiwillige werden gebeten, sich zu melden.]

Die StepGen-Darstellung ist der Softwarekomponente Stepgen nachempfunden. Die Standardeinstellung von StepGen ist ein aktiver High-Schrittausgang (High während der Schrittzeit (engl. step time), Low während des Schrittraums (engl. step space)). Um einen StepGen-Ausgangspin zu invertieren, wählen Sie den entsprechenden GPIO-Pin, der von StepGen verwendet wird. Um den GPIO-Pin zu finden, der für den StepGen-Ausgang verwendet wird, führen Sie *dmesg* wie oben gezeigt aus.

Jede StepGen-Instanz hat die folgenden Pins und Parameter:

Pins

- *control-type* - (bool, in) Switches between position control mode (0) and velocity control mode (1). Defaults to position control (0).
- *counts* - (sint, out) Feedback position in counts (number of steps).
- *enable* - (bool, in) Enables output steps. When false, no steps are generated.
- *position-cmd* - (real, in) Target position of stepper motion, in user-defined position units.
- *position-fb* - (real, out) Feedback position in user-defined position units (counts / position_scale).
- *velocity-cmd* - (real, in) Target velocity of stepper motion, in user-defined position units per second. This pin is only used when the stepgen is in velocity control mode (control-type=1).
- *velocity-fb* - (real, out) Feedback velocity in user-defined position units per second.

Parameter

- *dirhold* - (uint, rw) Minimum duration of stable Direction signal after a step ends, in nanoseconds.
- *dirsetup* - (uint, rw) Minimum duration of stable Direction signal before a step begins, in nanoseconds.
- *maxaccel* - (real, rw) Maximum acceleration, in position units per second per second. If set to 0, the driver will not limit its acceleration.
- *maxvel* - (real, rw) Maximum speed, in position units per second. If set to 0, the driver will choose the maximum velocity based on the values of steplen and stepspace (at the time that maxvel was set to 0).
- *position-scale* - (real, rw) Converts from counts to position units. position = counts / position_scale
- *step_type* - (uint, rw) Output format, like the step_type modparam to the software stepgen(9) component. 0 = Step/Dir, 1 = Up/Down, 2 = Quadrature. In Quadrature mode (step_type=2), the stepgen outputs one complete Gray cycle (00 -> 01 -> 11 -> 10 -> 00) for each *step* it takes.
- *steplen* - (uint, rw) Duration of the step signal, in nanoseconds.
- *stepspace* - (uint, rw) Minimum interval between step signals, in nanoseconds.

Ausgangsparameter

Die Step- und Direction-Pins der einzelnen StepGen haben zwei zusätzliche Parameter. Um herauszufinden, welcher I/O-Pin zu welchem Step- und Direction-Ausgang gehört, führen Sie *dmesg* wie oben beschrieben aus.

- *invert_output* - (bool, rw) This parameter only has an effect if the *is_output* parameter is true. If this parameter is true, the output value of the GPIO will be the inverse of the value on the *out* HAL pin.
- *is_opendrain* - (bool, rw) If this parameter is false, the GPIO behaves as a normal output pin: the I/O pin on the connector is driven to the value specified by the *out* HAL pin (possibly inverted). If this parameter is true, the GPIO behaves as an open-drain pin. Writing 0 to the *out* HAL pin drives the I/O pin low, writing 1 to the *out* HAL pin puts the I/O pin in a high-impedance state. In this high-impedance state the I/O pin floats (weakly pulled high), and other devices can drive the value; the resulting value on the I/O pin is available on the *in* and *in_not* pins. Only full GPIO pins and I/O pins used as outputs by active module instances have this parameter.

6.6.14. PWMGen

PWMgens haben Namen wie *hm2_<BoardType>.<BoardNum>.pwmgen.<Instance>*. *Instance* ist eine zweistellige Nummer, die der HostMot2 pwmgen Instanznummer entspricht. Es gibt *num_pwmgens* Instanzen, beginnend mit 00.

In HM2 verwendet jedes pwmgen drei Ausgangs-E/A-Pins: Not-Enable, Out0, und Out1. Um einen PWMGen-Ausgangspin zu invertieren, wählen Sie den entsprechenden GPIO-Pin, der von PWMGen verwendet wird. Um den GPIO-Pin zu finden, der für den PWMGen-Ausgang verwendet wird, führen Sie *dmesg* wie oben gezeigt aus.

Die Funktion der E/A-Pins Out0 und Out1 variiert je nach Ausgangstyp-Parameter (siehe unten).

Die hm2 pwmgen-Darstellung ist der Softwarekomponente pwmgen ähnlich. Jede pwmgen-Instanz hat die folgenden Pins und Parameter:

Pins

- *enable* - (bool, in) If true, the pwmgen will set its Not-Enable pin false and output its pulses. If *enable* is false, pwmgen will set its Not-Enable pin true and not output any signals.
- *value* - (real, in) The current pwmgen command value, in arbitrary units.

Parameter

- *output-type* - (sint, rw) This emulates the output_type load-time argument to the software pwmgen component. This parameter may be changed at runtime, but most of the time you probably want to set it at startup and then leave it alone. Accepted values are 1 (PWM on Out0 and Direction on Out1), 2 (Up on Out0 and Down on Out1), 3 (PDM mode, PDM on Out0 and Dir on Out1), and 4 (Direction on Out0 and PWM on Out1, *for locked antiphase*).
- *scale* - (real, rw) Scaling factor to convert *value* from arbitrary units to duty cycle: dc = value / scale. Duty cycle has an effective range of -1.0 to +1.0 inclusive, anything outside that range gets clipped.
- *pdm_frequency* - (uint, rw) This specifies the PDM frequency, in Hz, of all the pwmgen instances running in PDM mode (mode 3). This is the *pulse slot frequency*; the frequency at which the pdm generator in the Anything I/O board chooses whether to emit a pulse or a space. Each pulse (and space) in the PDM pulse train has a duration of 1/pdm_frequency seconds. For example, setting the pdm_frequency to $2*10^6$ Hz (2 MHz) and the duty cycle to 50% results in a 1 MHz square wave, identical to a 1 MHz PWM signal with 50% duty cycle. The effective range of this parameter is from about 1525 Hz up to just under 100 MHz. Note that the max frequency is determined by the ClockHigh frequency of the Anything I/O board; the 5I20 and 7I43 both have a 100 MHz clock, resulting in a 100 MHz max PDM frequency. Other boards may have different clocks, resulting in different max PDM frequencies. If the user attempts to set the frequency too high, then it will be clipped to the max supported frequency of the board.
- *pwm_frequency* - (uint, rw) This specifies the PWM frequency, in Hz, of all the pwmgen instances running in the PWM modes (modes 1 and 2). This is the frequency of the variable-duty-cycle wave. Its effective range is from 1 Hz up to 193 kHz. Note that the max frequency is determined by the ClockHigh frequency of the Anything I/O board; the 5i20 and 7i43 both have a 100 MHz clock, resulting in a 193 kHz max PWM frequency. Other boards may have different clocks, resulting in different max PWM frequencies. If the user attempts to set the frequency too high, then it will be clipped to the max supported frequency of the board. Frequencies below about 5 Hz are not terribly accurate, but above 5 Hz they are pretty close.

Ausgangsparameter

Die Ausgangspins der einzelnen PWMGen haben zwei zusätzliche Parameter. Um herauszufinden, welcher E/A-Pin zu welchem Ausgang gehört, führen Sie `dmesg` wie oben beschrieben aus.

- `invert_output` - (bool, rw) This parameter only has an effect if the `is_output` parameter is true. If this parameter is true, the output value of the GPIO will be the inverse of the value on the `out` HAL pin.

- `is_opendrain` - (bool, rw) If this parameter is false, the GPIO behaves as a normal output pin: The I/O pin on the connector is driven to the value specified by the `out` HAL pin (possibly inverted). If this parameter is true, the GPIO behaves as an open-drain pin. Writing 0 to the `out` HAL pin drives the I/O pin low, writing 1 to the `out` HAL pin puts the I/O pin in a high-impedance state. In this high-impedance state the I/O pin floats (weakly pulled high), and other devices can drive the value; the resulting value on the I/O pin is available on the `in` and `in_not` pins. Only full GPIO pins and I/O pins used as outputs by active module instances have this parameter.

6.6.15. Encoder

Encoder haben Namen wie `hm2_<BoardType>.<BoardNum>.encoder.<Instance>.`. `Instanz` ist eine zweistellige Zahl, die der Instanznummer des HostMot2-Encoders entspricht. Es gibt *num_encoders* Instanzen, beginnend mit 00.

Jeder Encoder verwendet drei oder vier Eingangs-I/O-Pins, je nachdem, wie die Firmware kompiliert wurde. Dreipolige Encoder verwenden A, B und Index (manchmal auch als Z bekannt). Vierpolige Encoder verwenden A, B, Index und Index-Maske.

Die hm2-Encoder-Darstellung ähnelt derjenigen, die von der kanonischen Geräteschnittstelle (im Dokument HAL General Reference) beschrieben wird, und der Software-Encoder-Komponente. Jede Encoder-Instanz hat die folgenden Pins und Parameter:

Pins

- `count` - (sint, out) Number of encoder counts since the previous reset.
- `index-enable` - (bool, i/o) When this pin is set to True, the count (and therefore also position) are reset to zero on the next Index (Phase-Z) pulse. At the same time, index-enable is reset to zero to indicate that the pulse has occurred.
- `position` - (real, out) Encoder position in position units (count / scale).
- `rawcounts` - (sint, out) Total number of encoder counts since the start, not adjusted for index or reset.
- `reset` - (bool, in) When this pin is TRUE, the count and position pins are set to 0. The value of the velocity pin is not affected by this. The driver does not reset this pin to FALSE after resetting the count to 0, that is the user's job.
- `velocity` - (real, out) Estimated encoder velocity in position units per second.

Parameter

- `counter-mode` - (bool, rw) Set to False (the default) for Quadrature. Set to True for Up/Down or for single input on Phase A. Can be used for a frequency to velocity converter with a single input on Phase A when set to true.
- `filter` - (bool, rw) If set to True (the default), the quadrature counter needs 15 clocks to register a change on any of the three input lines (any pulse shorter than this is rejected as noise). If set to False, the quadrature counter needs only 3 clocks to register a change. The encoder sample clock runs at 33 MHz on the PCI Anything I/O cards and 50 MHz on the 7I43.

- `index-invert` - (bool, rw) If set to True, the rising edge of the Index input pin triggers the Index event (if index-enable is True). If set to False, the falling edge triggers.
- `index-mask` - (bool, rw) If set to True, the Index input pin only has an effect if the Index-Mask input pin is True (or False, depending on the `index-mask-invert` pin below).
- `index-mask-invert` - (bool, rw) If set to True, Index-Mask must be False for Index to have an effect. If set to False, the `Index-Mask` pin must be True.
- `scale` - (real, rw) Converts from *count* units to *position* units. A quadrature encoder will normally have 4 counts per pulse so a 100 PPR encoder would be 400 counts per revolution. In `.counter-mode` a 100 PPR encoder would have 100 counts per revolution as it only uses the rising edge of A and direction is B.
- `vel-timeout` - (real, rw) When the encoder is moving slower than one pulse for each time that the driver reads the count from the FPGA (in the hm2_read() function), the velocity is harder to estimate. The driver can wait several iterations for the next pulse to arrive, all the while reporting the upper bound of the encoder velocity, which can be accurately guessed. This parameter specifies how long to wait for the next pulse, before reporting the encoder stopped. This parameter is in seconds.

6.6.16. 5I25 Konfiguration

Firmware

Die 5I25-Firmware wird für die Tochterkarte, mit der sie gekauft wurde, vorinstalliert. Die `firmware=xxx.BIT` ist also nicht Teil des hm2_pci-Konfigurationsstrings, wenn eine 5I25 verwendet wird.

Konfiguration

Beispielkonfigurationen der Karten 5I25/7I76 und 5I25/7I77 sind im Konfigurations-Auswahl enthalten.

Wenn Sie Ihre eigene Konfiguration erstellen möchten, zeigen die folgenden Beispiele, wie Sie die Treiber in die HAL-Datei laden.

5I25 + 7I76 Karte

```
# den generischen Treiber laden
loadrt hostmot2

# PCI-Treiber laden und konfigurieren
loadrt hm2_pci config="num_encoders=1 num_stepgens=5 sserial_port_0=0XXX"
```

5I25 + 7I77 Karte

```
# den generischen Treiber laden
loadrt hostmot2

# Laden Sie den PCI-Treiber und konfigurieren Sie ihn
loadrt hm2_pci config="num_encoders=6 num_pwmgens=6 sserial_port_0=0XXX"
```

SSERIAL-Konfiguration

Die Konfigurationszeichenfolge `sserial_port_0=0XXX` legt einige Optionen für die intelligente serielle Tochterkarte fest. Diese Optionen sind spezifisch für jede Tochterkarte. Weitere Informationen über die genaue Verwendung finden Sie im Mesa-Handbuch (normalerweise im Abschnitt SOFTWARE PROCESS DATA MODES) oder auf der Handbuchseite von SSERIAL(9).

7I77 Grenzwerte

Minlimit und Maxlimit sind Begrenzungen des Pin-Wertes (in diesem Fall des Analogausgangswertes), Fullscalemax ist der Skalierungsfaktor.

Diese sind standardmäßig auf den analogen Eingang oder den analogen Bereich (meist in Volt) eingestellt.

Bei den +-10 V-Analogausgängen der 7I77 sind die Standardwerte beispielsweise folgende:

minlimit: -10
maxlimit: +10
maxfullscale: 10

Wenn Sie z.B. den Analogausgang eines Kanals für ein Geschwindigkeitsservo auf IPS skalieren möchten (z.B. 24 IPS max), können Sie die Grenzen wie folgt festlegen:

minlimit: -24
maxlimit: +24
maxfullscale: 24

Wenn Sie das Analoge aus einem Kanal heraus auf Drehzahl für eine Spindel von 0 bis 6000 U/min mit 0-10 V Steuerung skalieren möchten, können Sie die Grenzen wie folgt festlegen:

minlimit: 0
maxlimit: 6000
maxfullscale: 6000
(dies würde verhindern, dass unerwünschte negative Ausgangsspannungen eingestellt werden)

6.6.17. Beispielkonfigurationen

Mehrere Beispielkonfigurationen für Mesa-Hardware sind in LinuxCNC enthalten. Die Konfigurationen sind in der hm2-servo und hm2-stepper Abschnitte der Konfigurations-Auswahl. In der Regel müssen Sie die Karte für die von Ihnen gewählte Konfiguration installieren, um sie zu laden. Die Beispiele sind ein guter Startpunkt und sparen Ihnen Zeit. Wählen Sie einfach das richtige Beispiel aus dem LinuxCNC Configuration Selector und speichern Sie eine Kopie auf Ihrem Computer, damit Sie es bearbeiten können. Um die genauen Pins und Parameter, die Ihre Konfiguration gab Ihnen, öffnen Sie das "Zeige HAL Konfiguration" (engl. Show HAL Configuration) Fenster aus dem Menü Maschine (engl. machine), oder führen Sie `dmesg` aus wie oben beschrieben.

6.7. MB2HAL

6.7.1. Einführung

MB2HAL ist eine generische nicht-Echtzeit HAL-Komponente zur Kommunikation mit einem oder mehreren Modbus-Geräten. Bislang gibt es zwei Möglichkeiten, mit einem Modbus-Gerät zu kommunizieren:

1. Eine Möglichkeit besteht darin, eine HAL-Komponente als Treiber zu erstellen, siehe VFD Modbus.
2. Eine andere Möglichkeit ist die Verwendung von Classic Ladder, das Modbus eingebaut hat, siehe ClassicLadder .
3. Jetzt gibt es eine dritte Option, die aus einem "generischen" Treiber besteht, der per Textdatei konfiguriert wird, dieser heißt MB2HAL.

Warum MB2HAL? Erwägen Sie den Einsatz von MB2HAL, wenn:

- Sie einen neuen Treiber schreiben müssenund keine Ahnung haben vom Programmieren.
- Sie müssen Classic Ladder "nur" verwenden, um die Modbus-Verbindungen zu verwalten.
- Sie müssen zunächst die Modbus-Transaktionen ermitteln und konfigurieren. MB2HAL hat Debug-Ebenen, um das Debuggen des Low-Level-Protokolls zu erleichtern.
- Sie haben mehr als ein Gerät zu verbinden. MB2HAL ist sehr effizient bei der Verwaltung mehrerer Geräte, Transaktionen und Verbindungen. Derzeit überwache ich zwei Achsentreiber über einen Rs232-Port, einen VFD-Treiber über einen anderen Rs232-Port und eine Remote I/O über TCP/IP.
- Sie wollen ein Protokoll, um Ihren Arduino mit HAL zu verbinden. Sehen Sie sich die mitgelieferte Beispiel-Konfigurationsdatei, Skizze und Bibliothek für Arduino Modbus an.

6.7.2. Safety Considerations

Modbus (RTU or TCP) is not suitable as the only run-enable path for a spindle or drive. A Modbus slave such as a VFD holds the last value written to its control registers indefinitely. If LinuxCNC exits, crashes, or the serial link drops while the spindle is running, no stop command is ever sent and the spindle keeps running. MB2HAL is a non-realtime userspace component and performs no cleanup on exit, so this applies even to a clean shutdown. Writing a stop command from `shutdown.hal` only covers the clean-exit case and is not a safety measure.

The standard practice is to layer the control:

1. Wire the enable in hardware. The VFD enable or STO (Safe Torque Off) inputs belong in the machine estop chain, independent of any communication bus. This is the only layer that qualifies as safety. Many VFDs offer dual redundant estop inputs for use with safety relays. With this layer in place, run and stop over Modbus is accepted practice; the remaining layers decide what happens when the communication fails.
2. Arrange a stop on communication loss. Most VFDs provide a Modbus timeout parameter with a fault or coast-to-stop action; set it to a few hundred milliseconds, so a crash or dropped link stops polling and the VFD stops the motor. Alternatively detect the loss in HAL: the `mb2hal.__m__.num_errors`

pin counts failed transactions (0 means the last transaction succeeded), and feeding it into a machine-off or estop chain stops the machine when the link dies. HAL-side detection avoids VFD-specific parameterization and needs no rework when the VFD is replaced, but coverage of a PC crash then rests on the output hardware watchdog, for example the Mesa hostmot2 watchdog or a parallel-port charge pump.

3. Optionally split the command sources. Configure run command from terminals and frequency reference from Modbus, drive the VFD run terminal from `motion.spindle-on` or `motion.machine-is-on` through a relay or digital output, and let Modbus carry only speed reference, direction and status. This also avoids the read-modify-write problem when a single control register multiplexes run, stop and other commands.

These considerations apply to every Modbus control path (MB2HAL, hm2_modbus, ClassicLadder or a custom driver), not only to MB2HAL.

6.7.3. Anwendung

a. Erstellen Sie eine Konfigurationsdatei nach folgendem Beispiel

   1. Komponentenname festlegen (optional)

      Set `HAL_MODULE_NAME=mymodule` (Voreinstellung `HAL_MODULE_NAME=mb2hal`)

   2. Laden der Modbus-HAL-nicht-Echtzeit-Komponente

b. Standard-Komponentenname: `loadusr -W mb2hal config=config_file.ini`

c. Benutzerdefinierter Komponentenname: `loadusr -Wn mymodule mb2hal config=config_file.ini`

6.7.4. Optionen

Init-Abschnitt

`[MB2HAL_INIT]`

Wert	Typ	Erforderlich	Beschreibung
`INIT_DEBUG`	Integer	Nein	Debug-Ebene der Init- und INI-Datei-Analyse. 0 = stumm 1 = Fehlermeldungen (Standard) 2 = OK-Bestätigungsmeldungen 3 = Debugging-Meldungen 4 = maximale Fehlersuchmeldungen (nur in Transaktionen)
`VERSION`	Zeichenfolge (engl. string)	Nein	Versionsnummer im Format N.N[NN]. Voreingestellt auf 1.0.

Wert	Typ	Erforderlich	Beschreibung
HAL_MODULE_NAME	Zeichenfolge (engl. string)	Nein	Name des HAL-Moduls (Komponente). Voreingestellt auf "mb2hal".
SLOWDOWN	Gleitkommazahl (engl. float)	Nein	Fügen Sie eine Verzögerung von "FLOAT Sekunden" zwischen Transaktionen ein, um eine umfangreiche Protokollierung zu vermeiden und die Fehlersuche zu erleichtern. Nützlich bei Verwendung von DEBUG=3 (NICHT INIT_DEBUG=3). Es betrifft ALLE Transaktionen. Verwenden Sie "0.0" für normale Aktivität.
TOTAL_TRANSACTIONS	Integer	Ja	Die Anzahl der gesamten Modbus-Transaktionen. Es gibt kein Maximum.

Transaktionsabschnitte

Für jede Transaktion ist ein Transaktionsabschnitt erforderlich, beginnend mit [TRANSACTION_00] und fortlaufend aufsteigend. Bei einer neuen Verknüpfung (nicht Transaktion) müssen Sie beim ersten Mal die Parameter REQUIRED angeben. Warnung: Alle nicht angegebenen OPTIONAL-Parameter werden von der vorherigen Transaktion übernommen.

Wert	Typ	Erforderlich	Beschreibung
LINK_TYPE	Zeichenfolge (engl. string)	Ja	Sie müssen entweder einen "serial" oder "tcp" Link für die erste Transaktion angeben. Spätere Transaktionen verwenden den vorherigen Transaktionslink, wenn er nicht angegeben.
TCP_IP	IP Adresse	If LINK_TYPE=tcp	Die Modbus-Slave IP-Adresse des Geräts. Wird ignoriert, wenn LINK_TYPE=serial.
TCP_PORT	Integer	Nein	Der TCP-Port des Modbus-Slave-Geräts. Der Standardwert ist 502. Wird ignoriert, wenn LINK_TYPE=serial.
SERIAL_PORT	Zeichenfolge (engl. string)	If LINK_TYPE=serial	Die serielle Schnittstelle. Zum Beispiel "/dev/ttyS0". Wird ignoriert, wenn LINK_TYPE=tcp.
SERIAL_BAUD	Integer	If LINK_TYPE=serial	Die Baudrate. Wird ignoriert, wenn LINK_TYPE=tcp.
SERIAL_BITS	Integer	If LINK_TYPE=serial	Datenbits. Eines von 5, 6, 7, 8. Wird ignoriert, wenn LINK_TYPE=tcp.

Wert	Typ	Erforderlich	Beschreibung
SERIAL_PARITY	Zeichenfolge (engl. string)	If LINK_TYPE=serial	Datenparität. Eine von: gerade, ungerade, keine. Wird ignoriert, wenn LINK_TYPE=tcp.
SERIAL_STOP	Integer	If LINK_TYPE=serial	Stoppbits. Eins von 1, 2. Wird ignoriert, wenn LINK_TYPE=tcp.
SERIAL_DELAY_MS	Integer	If LINK_TYPE=serial	Verzögerung des seriellen Ports ausschließlich zwischen Transaktionen in diesem Abschnitt. In ms. Standardwert ist 0. Ignoriert wenn LINK_TYPE=tcp.
MB_SLAVE_ID	Integer	Ja	Modbus-Slave-Nummer.
FIRST_ELEMENT	Integer	Ja	Die erste Elementadresse.
NELEMENTS	Integer	Sofern nicht PIN_NAMES angegeben ist	Die Anzahl der Elemente. Es ist ein Fehler, sowohl NELEMENTS als auch PIN_NAMES` ANZUGEBEN. Die Pin-Namen werden fortlaufende Nummern sein z.B. mb2hal.plcin.01.
PIN_NAMES	Liste	Sofern nicht NELEMENTS angegeben ist	Eine Liste von Elementnamen. Diese Namen werden für die Pin-Namen verwendet, z.B. mb2hal.plcin.cycle_start. **HINWEIS:** In der Liste dürfen keine Leerzeichen vorkommen. Beispiel: PIN_NAMES=cycle_start,stop,feed_hold
MB_TX_CODE	Zeichenfolge (engl. string)	Ja	Modbus Transaction Functions-Code (siehe Spezifikationen): • fnct_01_read_coils • fnct_02_read_discrete_inputs • fnct_03_read_holding_registers • fnct_04_read_input_registers • fnct_05_write_single_coil • fnct_06_write_single_register • fnct_15_write_multiple_coils • fnct_16_write_multiple_registers
MB_RESPONSE_TIMEOUT_MS	Integer	Nein	Antwort-Timeout für diese Transaktion. In ms. Der Standardwert ist 500 ms. Diese bestimmt, wie lange auf das 1. Byte gewartet werden soll, bevor ein Fehler ausgelöst wird.
MB_BYTE_TIMEOUT_MS	Integer	Nein	Byte-Timeout für diese Transaktion. In ms. Der Standardwert ist 500 ms. Dies bestimmt die Zeitspanne, die von Byte zu Byte warten gewartet wird, bevor ein Fehler ausgelöst wird.

Wert	Typ	Erforderlich	Beschreibung
HAL_TX_NAME	Zeichenfolge (engl. string)	Nein	Anstatt die Transaktions-Nummer anzugeben, verwenden Sie einen Namen. Beispiel: mb2hal.00.01 könnte zu mb2hal.plcin.01 werden. Der Name darf nicht länger als 28 Zeichen sein. **HINWEIS:** Achten Sie bei der Verwendung von Namen darauf, dass Sie nicht zwei Transaktionen mit demselben Namen erhalten.
MAX_UPDATE_RATE	Gleitkommazahl (engl. float)	Nein	Maximale Aktualisierungsrate in Hz. Standardwert ist 0.0 (0.0 = so schnell wie möglich = unendlich). **HINWEIS:** Dies ist eine maximale Rate, die tatsächliche Rate kann niedriger sein. Wenn Sie sie in ms berechnen wollen, verwenden Sie (1000 / required_ms). Beispiel: 100 ms = MAX_UPDATE_RATE=10.0, denn 1000,0 ms / 100,0 ms = 10,0 Hz.
DEBUG	Zeichenfolge (engl. string)	Nein	Debug-Level nur für diese Transaktion. Siehe Parameter INIT_DEBUG oben.

Fehlercodes

Beachten Sie beim Debuggen von Transaktionen, dass der zurückgegebene Wert "ret[]" entspricht:

Ausnahmen vom Modbus-Protokoll:

- 0x01 - ILLEGAL_FUNCTION - der in der Abfrage empfangene FUNCTION-Code ist nicht erlaubt oder ungültig.
- 0x02 - ILLEGAL_DATA_ADDRESS - die in der Abfrage empfangene DATA ADDRESS ist keine zulässige Adresse für den Slave oder ist ungültig.
- 0x03 - ILLEGAL_DATA_VALUE - ein im Datenabfragefeld enthaltener WERT ist kein zulässiger Wert oder ist ungültig.
- 0x04 - SLAVE_DEVICE_FAILURE - Nicht wiederherstellbarer Fehler des SLAVE- (oder MASTER-) Geräts beim Versuch, die angeforderte Aktion durchzuführen.
- 0x04 - SERVER_FAILURE - (siehe oben).
- 0x05 - ACKNOWLEDGE - Diese Antwort wird zurückgegeben, um einen TIMEOUT im Master zu VERMEIDEN. Für die Bearbeitung der Anfrage im Slave ist eine lange Zeitspanne erforderlich.
- 0x06 - SLAVE_DEVICE_BUSY - Der Slave (oder Server) ist BUSY. Übertragen Sie die Anfrage später erneut.
- 0x06 - SERVER_BUSY - (siehe oben).
- 0x07 - NEGATIVE_ACKNOWLEDGE - Erfolglose Programmieranfrage mit Funktionscode 13 oder 14.
- 0x08 - MEMORY_PARITY_ERROR - SLAVE-Paritätsfehler im SPEICHER.
- 0x0A (-10) - GATEWAY_PROBLEM_PATH - Gateway-Pfad(e) nicht verfügbar.

- 0x0B (-11) - GATEWAY_PROBLEM_TARGET - Das Zielgerät hat nicht geantwortet (vom Master, nicht vom Slave erzeugt).

Programm oder Verbindung:

- 0x0C (-12) - COMM_TIME_OUT
- 0x0D (-13) - PORT_SOCKET_FAILURE
- 0x0E (-14) - SELECT_FAILURE
- 0x0F (-15) - TOO_MANY_DATAS
- 0x10 (-16) - INVALID_CRC
- 0x11 (-17) - INVALID_EXCEPTION_CODE

6.7.5. Beispiel Konfigurationsdatei

Klicke hier zum Herunterladen.

```
#This .INI file is also the HELP, MANUAL and HOW-TO file for mb2hal.

#Load the Modbus HAL userspace module as the examples below,
#change to match your own HAL_MODULE_NAME and INI file name
#Using HAL_MODULE_NAME=mb2hal or nothing (default): loadusr -W mb2hal
config=config_file.ini
#Using HAL_MODULE_NAME=mymodule: loadusr -Wn mymodule mb2hal config=config_file.ini

# ++++++++++++++++++++++++
# Common section
# ++++++++++++++++++++++++
[MB2HAL_INIT]

#OPTIONAL: Debug level of init and INI file parsing.
# 0 = silent.
# 1 = error messages (default).
# 2 = OK confirmation messages.
# 3 = debugging messages.
# 4 = maximum debugging messages (only in transactions).
INIT_DEBUG=3

#OPTIONAL: Set to 1.1 to enable the new functions:
# - fnct_01_read_coils
# - fnct_05_write_single_coil
# - changed pin names (see https://linuxcnc.org/docs/2.9/html/drivers/mb2hal.html#_pins).
VERSION=1.1

#OPTIONAL: HAL module (component) name. Defaults to "mb2hal".
HAL_MODULE_NAME=mb2hal

#OPTIONAL: Insert a delay of "FLOAT seconds" between transactions in order
#to not to have a lot of logging and facilitate the debugging.
#Useful when using DEBUG=3 (NOT INIT_DEBUG=3)
#It affects ALL transactions.
#Use "0.0" for normal activity.
```

```
SLOWDOWN=0.0

#REQUIRED: The number of total Modbus transactions. There is no maximum.
TOTAL_TRANSACTIONS=9

# ++++++++++++++++++++++++
# Transactions
# ++++++++++++++++++++++++
#One transaction section is required per transaction, starting at 00 and counting up
sequentially.
#If there is a new link (not transaction), you must provide the REQUIRED parameters 1st
time.
#Warning: Any OPTIONAL parameter not specified are copied from the previous transaction.
[TRANSACTION_00]

#REQUIRED: You must specify either a "serial" or "tcp" link for the first transaction.
#Later transaction will use the previous transaction link if not specified.
LINK_TYPE=tcp

#if LINK_TYPE=tcp then REQUIRED (only 1st time): The Modbus slave device ip address.
#if LINK_TYPE=serial then IGNORED
TCP_IP=192.168.2.10

#if LINK_TYPE=tcp then OPTIONAL.
#if LINK_TYPE=serial then IGNORED
#The Modbus slave device tcp port. Defaults to 502.
TCP_PORT=502

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#The serial port.
SERIAL_PORT=/dev/ttyS0

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#The baud rate.
SERIAL_BAUD=115200

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Data bits. One of 5,6,7,8.
SERIAL_BITS=8

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Data parity. One of: even, odd, none.
SERIAL_PARITY=none

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Stop bits. One of 1, 2.
SERIAL_STOP=2

#if LINK_TYPE=serial then OPTIONAL:
#if LINK_TYPE=tcp then IGNORED
#Serial port delay between for this transaction only.
```

```
#In ms. Defaults to 0.
SERIAL_DELAY_MS=10

#REQUIRED (only 1st time).
#Modbus slave number.
MB_SLAVE_ID=1

#REQUIRED: The first element address (decimal integer).
FIRST_ELEMENT=0

#REQUIRED unless PIN_NAMES is specified: The number of elements.
#It is an error to specify both NELEMENTS and PIN_NAMES
#The pin names will be sequential numbers e.g mb2hal.plcin.01
#NELEMENTS=4

#REQUIRED unless NELEMENTS is specified: A list of element names.
#these names will be used for the pin names, e.g mb2hal.plcin.cycle_start
#NOTE: there must be no white space characters in the list
PIN_NAMES=cycle_start,stop,feed_hold

#REQUIRED: Modbus transaction function code (see www.modbus.org specifications).
#    fnct_01_read_coils               (01 = 0x01) (new in 1.1)
#    fnct_02_read_discrete_inputs     (02 = 0x02)
#    fnct_03_read_holding_registers   (03 = 0x03)
#    fnct_04_read_input_registers     (04 = 0x04)
#    fnct_05_write_single_coil        (05 = 0x05) (new in 1.1)
#    fnct_06_write_single_register    (06 = 0x06)
#    fnct_15_write_multiple_coils     (15 = 0x0F)
#    fnct_16_write_multiple_registers (16 = 0x10)
#
# Created pins:
# fnct_01_read_coils:
# fnct_02_read_discrete_inputs:
#     mb2hal.m.n.bit     (output)
#     mb2hal.m.n.bit-inv (output)
# fnct_03_read_holding_registers:
# fnct_04_read_input_registers:
#     mb2hal.m.n.float   (output)
#     mb2hal.m.n.int     (output)
# fnct_05_write_single_coil:
#     mb2hal.m.n.bit     (input)
#     NELEMENTS needs to be 1 or PIN_NAMES must contain just one name.
# fnct_06_write_single_register:
#     mb2hal.m.n.float   (input)
#     mb2hal.m.n.int     (input)
#     NELEMENTS needs to be 1 or PIN_NAMES must contain just one name.
#     Both pin values are added and limited to 65535 (UINT16_MAX). Normally use one and
let the other open (read as 0).
# fnct_15_write_multiple_coils:
#     mb2hal.m.n.bit     (input)
# fnct_16_write_multiple_registers:
#     mb2hal.m.n.float   (input)
#     mb2hal.m.n.int     (input)
#     Both pin values are added and limited to 65535 (UINT16_MAX). Normally use one and
let the other open (read as 0).
#
```

```
# m = HAL_TX_NAME or transaction number if not set, n = element number (NELEMENTS) or
name from PIN_NAMES
# Example: mb2hal.00.01.<type> (transaction=00, second register=01 (00 is the first one))
#          mb2hal.TxName.01.<type> (HAL_TX_NAME=TxName, second register=01 (00 is the
first one))
MB_TX_CODE=fnct_03_read_holding_registers

#OPTIONAL: Response timeout for this transaction. In INTEGER ms. Defaults to 500 ms.
#This is how much to wait for 1st byte before raise an error.
MB_RESPONSE_TIMEOUT_MS=500

#OPTIONAL: Byte timeout for this transaction. In INTEGER ms. Defaults to 500 ms.
#This is how much to wait from byte to byte before raise an error.
MB_BYTE_TIMEOUT_MS=500

#OPTIONAL: Instead of giving the transaction number, use a name.
#Example: mb2hal.00.01 could become mb2hal.plcin.01
#The name must not exceed 28 characters.
#NOTE: when using names be careful that you dont end up with two transactions
#using the same name.
HAL_TX_NAME=remoteIOcfg

#OPTIONAL: Maximum update rate in HZ. Defaults to 0.0 (0.0 = as soon as available =
infinite).
#NOTE: This is a maximum rate and the actual rate may be lower.
#If you want to calculate it in ms use (1000 / required_ms).
#Example: 100 ms = MAX_UPDATE_RATE=10.0, because 1000.0 ms / 100.0 ms = 10.0 Hz
MAX_UPDATE_RATE=0.0

#OPTIONAL: Debug level for this transaction only.
#See INIT_DEBUG parameter above.
DEBUG=2

#While DEBUGGING transactions note the returned "ret[]" value correspond to:
#/* Modbus protocol exceptions */
#ILLEGAL_FUNCTION        -0x01 the FUNCTION code received in the query is not allowed or
invalid.
#ILLEGAL_DATA_ADDRESS    -0x02 the DATA ADDRESS received in the query is not an allowable
address for the slave or is invalid.
#ILLEGAL_DATA_VALUE      -0x03 a VALUE contained in the data query field is not an
allowable value or is invalid.
#SLAVE_DEVICE_FAILURE    -0x04 SLAVE (or MASTER) device unrecoverable FAILURE while
attempting to perform the requested action.
#SERVER_FAILURE          -0x04 (see above).
#ACKNOWLEDGE             -0x05 This response is returned to PREVENT A TIMEOUT in the
master.
#                              A long duration of time is required to process the request
in the slave.
#SLAVE_DEVICE_BUSY       -0x06 The slave (or server) is BUSY. Retrasmit the request
later.
#SERVER_BUSY             -0x06 (see above).
#NEGATIVE_ACKNOWLEDGE    -0x07 Unsuccessful programming request using function code 13 or
14.
#MEMORY_PARITY_ERROR     -0x08 SLAVE parity error in MEMORY.
#GATEWAY_PROBLEM_PATH    -0x0A (-10) Gateway path(s) not available.
#GATEWAY_PROBLEM_TARGET  -0x0B (-11) The target device failed to respond (generated by
```

```
master, not slave).
#/* Program or connection */
#COMM_TIME_OUT           -0x0C (-12)
#PORT_SOCKET_FAILURE     -0x0D (-13)
#SELECT_FAILURE          -0x0E (-14)
#TOO_MANY_DATAS          -0x0F (-15)
#INVALID_CRC             -0x10 (-16)
#INVALID_EXCEPTION_CODE  -0x11 (-17)

[TRANSACTION_01]
MB_TX_CODE=fnct_01_read_coils
FIRST_ELEMENT=1024
NELEMENTS=24
HAL_TX_NAME=remoteIOin
MAX_UPDATE_RATE=0.0
DEBUG=1

[TRANSACTION_02]
MB_TX_CODE=fnct_02_read_discrete_inputs
FIRST_ELEMENT=1280
NELEMENTS=8
HAL_TX_NAME=readStatus
MAX_UPDATE_RATE=0.0

[TRANSACTION_03]
MB_TX_CODE=fnct_05_write_single_coil
FIRST_ELEMENT=100
NELEMENTS=1
HAL_TX_NAME=setEnableout
MAX_UPDATE_RATE=0.0

[TRANSACTION_04]
MB_TX_CODE=fnct_15_write_multiple_coils
FIRST_ELEMENT=150
NELEMENTS=10
HAL_TX_NAME=remoteIOout
MAX_UPDATE_RATE=0.0

[TRANSACTION_05]
LINK_TYPE=serial
SERIAL_PORT=/dev/ttyS0
SERIAL_BAUD=115200
SERIAL_BITS=8
SERIAL_PARITY=none
SERIAL_STOP=2
SERIAL_DELAY_MS=50
MB_SLAVE_ID=1
MB_TX_CODE=fnct_03_read_holding_registers
FIRST_ELEMENT=1
NELEMENTS=2
HAL_TX_NAME=XDrive01
MAX_UPDATE_RATE=0.0
DEBUG=1

[TRANSACTION_06]
MB_TX_CODE=fnct_04_read_input_registers
```

```
FIRST_ELEMENT=12
NELEMENTS=3
HAL_TX_NAME=XDrive02
MAX_UPDATE_RATE=10.0
DEBUG=1

[TRANSACTION_07]
MB_TX_CODE=fnct_06_write_single_register
FIRST_ELEMENT=20
NELEMENTS=1
HAL_TX_NAME=XDrive03
MAX_UPDATE_RATE=0.0
DEBUG=1

[TRANSACTION_08]
MB_TX_CODE=fnct_16_write_multiple_registers
FIRST_ELEMENT=55
NELEMENTS=8
HAL_TX_NAME=XDrive04
MAX_UPDATE_RATE=10.0
DEBUG=1
```

6.7.6. Pins

NOTE

Yellow = Neu in MB2HAL 1.1 (LinuxCNC 2.9) Um diese neuen Funktionen zu nutzen, müssen Sie `VERSION = 1.1` setzen.

m = Wert von `HAL_TX_NAME` wenn gesetzt, sonst Transaktionsnummer
n = Elementnummer (`NELEMENTS`) oder Name aus `PIN_NAMES`

Beispiel:

- `mb2hal.00.01.int` (TRANSACTION_00, zweites Register)
- `mb2hal.readStatus.00.bit` (HAL_TX_NAME=readStatus, first bit)

fnct_01_read_coils

- mb2hal.*m*.*n*.bit *bool out*
- mb2hal.*m*.*n*.bit-inv *bool out*

fnct_02_read_discrete_inputs

- mb2hal.*m*.*n*.bit *bool out*
- mb2hal.*m*.*n*.bit-inv *bool out*

fnct_03_read_holding_registers

- mb2hal.*m*.*n*.float *real out*
- mb2hal.*m*.*n*.int *sint out*

fnct_04_read_input_registers

- mb2hal.*m.n.*float *real out*
- mb2hal.*m.n.*int *sint out*

fnct_05_write_single_coil

- mb2hal.*m.n.*bit *bool in*

`NELEMENTS` muss 1 sein oder `PIN_NAMES` darf nur einen Namen enthalten.

fnct_06_write_single_register

- mb2hal.*m.n.*float *real in*
- mb2hal.*m.n.*int *sint in*

`NELEMENTS` muss 1 sein oder `PIN_NAMES` darf nur einen Namen enthalten. Beide Pinwerte werden addiert und auf 65535 (UINT16_MAX) begrenzt. Verwenden Sie einen und lassen Sie den anderen offen (wird als 0 gelesen).

fnct_15_write_multiple_coils

- mb2hal.*m.n.*bit *bool in*

fnct_16_write_multiple_registers

- mb2hal.*m.n.*float *real in*
- mb2hal.*m.n.*int *sint in*

Beide Pin-Werte werden addiert und auf 65535 (UINT16_MAX) begrenzt. Verwenden Sie einen und lassen Sie den anderen offen (gelesen als 0).

6.8. Mitsub VFD-Treiber

Dies ist ein nicht-Echtzeit-HAL-Programm, geschrieben in Python, zur Steuerung von VFDs von Mitsubishi.
Speziell die A500 F500 E500 A500 D700 E700 F700 Serie - andere können funktionieren.
mitsub_vfd unterstützt die serielle Steuerung über das RS485-Protokoll.
Die Konvertierung von USB oder serieller Schnittstelle zu RS485 erfordert spezielle Hardware.

NOTE Da es sich um ein nicht-Echtzeit-Programm handelt, kann es durch Computerbelastung und Latenzzeiten beeinträchtigt werden. Es ist möglich, die Kontrolle über die VFDs zu verlieren. Es ist optional möglich, den VFD so einzustellen, dass er bei Verlust der Kommunikation stoppt, falls dies gewünscht wird. Es sollte immer eine Notaus-Schaltung vorhanden sein, die im Notfall die Stromzufuhr zum Gerät unterbricht.

Diese Komponente wird mit dem halcmd-Befehl "loadusr" geladen:

```
loadusr -Wn coolant mitsub_vfd spindle=02 coolant=01
```

Der obige Befehl lautet:
loadusr, warten bis Kühlmittelpins bereit sind, Komponente mitsub_vfd, mit 2 Slaves namens Spindel (Slave #2) und Kühlmittel (Slave #1)

6.8.1. Kommandozeilen-Optionen

Die Kommandozeilenoptionen sind:

- *-b oder --baud <rate>* : die Baudrate einstellen - muss für alle vernetzten VFDs gleich sein
- *-p oder --port <device path>* : legt den zu verwendenden Anschluss fest, z. B. /dev/ttyUSB0
- *<name>=<slave#>* : setzt den Namen der HAL-Komponente/des Pins und die Slave-Nummer.

Das Debugging kann durch Setzen des Debug-Pins auf true umgeschaltet werden.

NOTE Das Einschalten der Fehlersuche (engl. debugging) führt zu einer Flut von Text im Terminal.

6.8.2. Pins

Dabei steht <n> für `mitsub_vfd` oder den beim Laden vergebenen Namen.

- *<n>.fwd* (bool, in) True sets motion forward, False sets reverse.
- *<n>.run* (bool, in) True sets the VFD in motion based on the .fwd pin.
- *<n>.debug* (bool, in) Prints debug info to the terminal.
- *<n>.alarm* (bool, out) signals an alarm state of VFD.
- *<n>.up-to-speed* (bool, out) when drive is at commanded speed (speed-tolerance is set on vfd)
- *<n>.monitor* (bool, in) some models (eg E500) cannot monitor status - set the monitor pin to false in this case pins such as up-to-speed, amps, alarm and status bits are not updated.
- *<n>.motor-cmd* (real, in) commanded speed to the VFD (scaled to hertz by default).
- *<n>.motor-fb* (real, out) feedback speed from the VFD (scaled to hertz by default).
- *<n>.motor-amps* (real, out) Current amperage output of motor.
- *<n>.motor-power* (real, out) Current power output of motor.
- *<n>.scale-cmd* (real, in) Scales the motor-cmd pin to arbitrary units. default 1 = Hertz.
- *<n>.scale-fb* (real, in) Scales the motor-fb pin to arbitrary units. default 1 = Hertz.
- *<n>.scale-amps* (real, in) Scales the motor-amps pin to arbitrary units. default 1 = amps.
- *<n>.scale-power* (real, in) Scales the motor-power pin to arbitrary units. default 1 = .
- *<n>.estop* (bool, in) puts the VFD into emergency-stopped status.
- *<n>.stat-bit-N* (bool, out) N = 0 to 7, status bits are user configurable on the VFD. Bit 3 should be set to

at speed and bit 7 should be set to alarm. Others are free to be set as required.

6.8.3. HAL-Beispiel

```
#
# Beispiel für die Verwendung des Mitsubishi VFD-Treibers
#
loadusr -Wn coolant mitsub_vfd spindle=02 coolant=01

**************** Spindle VFD setup slave 2 *********************
net spindel-vel-cmd spindel.motor-cmd
net spindel-cw spindel.fwd
net spindel-on spindel.run
net spindel-at-speed spindel.up-to-speed
net estop-out spindel.estop
# cmd skaliert auf RPM
setp spindel.scale-cmd .135
# Rückmeldung erfolgt in U/min
setp spindel.scale-fb 7.411
# ermöglicht es uns, den Status zu sehen
setp spindel.monitor 1

net spindle-speed-indicator spindle.motor-fb        gladevcp.spindle-speed

*************** Kühlmittel vfd setup slave 3 ***********************
net coolant-flood               coolant.run
net coolant-is-on               coolant.up-to-speed   gladevcp.coolant-on-led
net estop-out                   coolant.estop
# cmd und Rückmeldung skaliert auf Hertz
setp coolant.scale-cmd 1
setp coolant.scale-fb
# Befehl volle Geschwindigkeit
setp coolant.motor-cmd 60
# ermöglicht die Anzeige des Status
setp coolant.monitor 1
```

6.8.4. Konfigurieren des Mitsubishi VFD für die serielle Nutzung

Anschließen der seriellen Schnittstelle

Die Mitsubishi VFDs haben eine RJ-45-Buchse für die serielle Kommunikation.
Da sie das RS485-Protokoll verwenden, können sie Punkt-zu-Punkt miteinander vernetzt werden.
Dieser Treiber wurde mit dem Opto22 AC7A getestet, um von RS232 auf RS485 zu konvertieren.

Modbus-Einrichtung

Referenzhandbücher:

Referenzhandbuch für Kommunikationsoptionen" und "Technisches Handbuch A500" für die Serie 500.
Technisches Handbuch "Fr-A700 F700 E700 D700" für die Serie 700

Die PR-Einstellungen des VFD müssen für die serielle Kommunikation manuell angepasst werden.
Man muss den VFD einschalten, damit einige dieser Einstellungen registriert werden, z. B. PR 79

- *PR 77* auf 1 gesetzt *-um andere PR-Änderungen freizuschalten.*
- *PR 79* auf 1 oder 0 gesetzt '-für die Kommunikation über die serielle Schnittstelle '
- *PR 117* auf 0-31 gesetzt *-Slave-Nummer, Treiber muss auf dieselbe Nummer verweisen.*
- *PR 118* getestet mit 96 *-Baudrate (kann auf 48,96,192 eingestellt werden), wenn auch der Treiber eingestellt ist.*
- *PR 119* auf 0 gesetzt *-Stoppbit/Datenlänge (8 Bits, zwei Stopps)*
- PR 120' auf 0 gesetzt *-keine Parität*
- *PR 121* eingestellt auf 1-10 '-wenn 10 (maximal) COM-Fehler, dann VFD-Fehler '
- *PR 122* getestet mit 9999 '-wenn die Kommunikation verloren geht, hat der VFD keinen Fehler '
- *PR 123* auf 9999 eingestellt *-dem seriellen Datenrahmen wird keine Wartezeit hinzugefügt.*
- *PR 124* auf 0 gesetzt *-kein Zeilenumbruch am Ende der Zeile.*

6.9. Pico-Treiber

Pico Systems hat eine Familie von Karten für analoge Servo-, Stepper- und PWM (digitale) Servosteuerung entwickelt. Die Karten werden über eine parallele Schnittstelle an den PC angeschlossen und arbeiten im EPP-Modus. Obwohl die meisten Benutzer nur eine Karte an einen Parallelport anschließen, kann theoretisch jede beliebige Kombination von bis zu 8 oder 16 Karten an einem einzigen Parallelport verwendet werden. Ein Treiber bedient alle Arten von Karten. Die endgültige Mischung der E/A hängt von der/den angeschlossenen Karte(n) ab. Der Treiber unterscheidet nicht zwischen den Karten, er nummeriert einfach die E/A-Kanäle (Encoder usw.), beginnend mit 0 auf der ersten Karte. Der Treiber heißt hal_ppmc.ko Die analoge Servo-Schnittstelle wird auch PPMC für Parallel Port Motion Control genannt. Es gibt auch den Universal Stepper Controller, abgekürzt USC. Und den Universal PWM Controller, abgekürzt UPC.

Installation:

```
loadrt hal_ppmc port_addr=<addr1>[,<addr2>[,<addr3>...]]
```

Der Parameter *port_addr* teilt dem Treiber mit, welche parallele(n) Schnittstelle(n) dieser überprüfen soll. Standardmäßig ist <*addr1*> 0x0378, und <*addr2*> und folgende werden nicht verwendet. Der Treiber durchsucht den gesamten Adressraum der erweiterten parallelen Schnittstelle(n) unter *port_addr* und sucht nach einer oder mehreren Karten der PPMC-Familie. Es exportiert dann HAL-Pins für alles, was es findet. Während des Ladens (oder des versuchten Ladens) gibt der Treiber einige nützliche Debugging-Meldungen in das Kernel-Log aus, die mit *dmesg* eingesehen werden können.

Es können bis zu 3 Parport-Busse verwendet werden, und jeder Bus kann bis zu 8 (oder möglicherweise 16 PPMC) Geräte enthalten.

6.9.1. Kommandozeilen-Optionen

In der Befehlszeile von loadrt können mehrere Optionen angegeben werden. Erstens kann bei USC und UPC ein 8-Bit-DAC für die Spindeldrehzahlsteuerung und ähnliche Funktionen hinzugefügt werden. Dies kann mit dem Parameter extradac=*0xnn[,0xmm]* angegeben werden. Mit dem in [ ] eingeschlossenen Teil können Sie diese Option auf mehr als einer Platine des Systems angeben. Die erste Hexadezimalziffer gibt an, auf welchen EPP-Bus Bezug genommen wird; sie entspricht der Reihenfolge der Portadressen im Parameter port_addr, wobei <addr1> hier Null wäre. Für den ersten EPP-Bus würde die erste USC- oder UPC-Platine also mit *0x00* beschrieben, die zweite USC- oder UPC-Platine auf demselben Bus wäre *0x02*. (Beachten Sie, dass jede USC- oder UPC-Platine zwei Adressen belegt, wenn also eine auf 00 steht, müsste die nächste 02 sein.)

Alternativ können die 8 digitalen Ausgangspins als zusätzliche digitale Ausgänge verwendet werden, es funktioniert genauso wie oben mit der Syntax : extradout=0xnn'. Die Extradac- und Extradout-Optionen schließen sich auf jedem Board gegenseitig aus, Sie können nur eine angeben.

Die Encoderplatinen UPC und PPMC können das Eintreffen von Encoderzählungen mit einem Zeitstempel versehen, um die Ableitung der Achsengeschwindigkeit zu verfeinern. Diese abgeleitete Geschwindigkeit kann in die PID hal-Komponente eingespeist werden, um eine glattere D-Term-Reaktion zu erzeugen. Die Syntax lautet: timestamp=*0xnn[,0xmm]*, dies funktioniert auf die gleiche Weise wie oben, um auszuwählen, welche Karte konfiguriert werden soll. Standardmäßig ist die Option timestamp nicht aktiviert. Wenn Sie diese Option in die Befehlszeile eingeben, wird die Option aktiviert. Das erste *n* wählt den EPP-Bus aus, das zweite entspricht der Adresse der Karte, auf der die Option aktiviert ist. Der Treiber prüft den Revisionsstand der Karte, um sicherzustellen, dass die Firmware die Funktion unterstützt, und gibt eine Fehlermeldung aus, wenn die Karte sie nicht unterstützt.

Die PPMC-Geberkarte verfügt über eine Option zur Auswahl der Frequenz des digitalen Geberfilters. (Die UPC hat die gleiche Möglichkeit über DIP-Schalter auf der Karte.) Da die PPMC-Geberkarte diese zusätzlichen DIP-Schalter nicht hat, muss sie über eine Kommandozeilenoption ausgewählt werden. Standardmäßig läuft der Filter mit 1 MHz, so dass Encoder bis zu etwa 900 kHz gezählt werden können (abhängig von Rauschen und Quadraturgenauigkeit des Encoders). Die Optionen sind 1, 2,5, 5 und 10 MHz. Diese werden mit einem Parameter von 1, 2, 5 und 10 (dezimal) eingestellt, der als Hexadezimalziffer "A" angegeben wird. Diese werden in ähnlicher Weise wie die obigen Optionen angegeben, wobei die Frequenzeinstellung links von den Bus-/Adressziffern steht. Um also 5 MHz auf der Encoderplatine an Adresse 3 auf dem ersten EPP-Bus einzustellen, würden Sie schreiben: `enc_clock=*0x503* `.

Vor kurzem wurde festgestellt, dass einige Parallelport-Chips nicht mit dem ppmc-Treiber funktionieren. Insbesondere der Oxford OXPCIe952 Chip auf den SIIG PCIe Parallelport-Karten hatte dieses Problem. Der ppmc-Treiber in allen LinuxCNC-Versionen ab 2.7.8 sind für dieses Problem standardmäßig korrigiert worden. Allerdings könnte dies möglicherweise zu Problemen mit wirklich alten EPP Parallelport-Hardware, so gibt es eine Kommandozeilen-Option, um wieder auf das vorherige Verhalten zu gehen. Das neue Verhalten wird standardmäßig eingestellt, oder durch Hinzufügen des Parameters epp_dir=0 auf der Befehlszeile. Um das alte Verhalten zu erhalten, fügen Sie epp_dir=1 in die Befehlszeile ein. Alle parallelen Ports, die ich hier habe, arbeiten mit dem neuen Standardverhalten. Wie bei den anderen Parametern ist es möglich, eine Liste anzugeben, wie z.B. epp_dir=1,0,1, um verschiedene Einstellungen für jede der bis zu 3 parallelen Schnittstellen zu setzen.

6.9.2. Pins

Bei den folgenden Pins, Parametern und Funktionen ist <port> die ID der parallelen Schnittstelle. Gemäß den Namenskonventionen sollte der erste Port immer eine ID von Null haben. Alle Karten verfügen über eine Methode zur Einstellung der Adresse auf dem EPP-Bus. USC und UPC haben einfache Vorkehrungen für nur zwei Adressen, aber durch Jumper-Folienschnitte können bis zu 4 Karten adressiert werden. Die PPMC-Karten haben 16 mögliche Adressen. In allen Fällen zählt der Treiber die Karten nach Typ auf und exportiert die entsprechenden HAL-Pins. Zum Beispiel werden die Encoder von Null aufwärts aufgezählt, in der gleichen Reihenfolge, wie die Adressschalter auf der Karte angeben. Die erste Platine hat also die Encoder 0 bis 3, die zweite Platine die Encoder 4 bis 7. Die erste Spalte nach dem Aufzählungspunkt gibt an, welche Platinen mit diesem HAL-Pin oder Parameter verbunden sind. Alle bedeutet, dass dieser Pin auf allen drei Platinentypen verfügbar ist. Option bedeutet, dass dieser Pin nur exportiert wird, wenn diese Option durch einen optionalen Parameter im loadrt HAL-Befehl aktiviert ist. Diese Optionen setzen voraus, dass die Platine einen ausreichenden Revisionsstand hat, um die Funktion zu unterstützen.

- *(All sint output) ppmc.<port>.encoder.<channel>.count* - Encoder position, in counts.
- *(All sint output) ppmc.<port>.encoder.<channel>.delta* - Change in counts since last read, in raw encoder count units.
- *(All real output) 'ppmc.<port>.encoder.<channel>.velocity* - Velocity scaled in user units per second. On PPMC and USC this is derived from raw encoder counts per servo period, and hence is affected by encoder granularity. On UPC boards with the 8/21/09 and later firmware, velocity estimation by timestamping encoder counts can be used to improve the smoothness of this velocity output. This can be fed to the PID HAL component to produce a more stable servo response. This function has to be enabled in the HAL command line that starts the PPMC driver, with the timestamp=0x00 option.
- *(All real output) ppmc.<port>.encoder.<channel>.position* - Encoder position, in user units.
- *(All bool bidir) ppmc.<port>.encoder.<channel>.index-enable* - Connect to joint.#.index-enable for home-to-index. This is a bidirectional HAL signal. Setting it to true causes the encoder hardware to reset the count to zero on the next encoder index pulse. The driver will detect this and set the signal back to false.
- *(PPMC real input) ppmc.<port>.DAC.<channel>.value* - sends a signed value to the 16-bit Digital to Analog Converter on the PPMC DAC16 board commanding the analog output voltage of that DAC channel.
- *(UPC bool input) ppmc.<port>.pwm.<channel>.enable* - Enables a PWM generator.
- *(UPC real input) ppmc.<port>.pwm.<channel>.value* - Value which determines the duty cycle of the PWM waveforms. The value is divided by *pwm.<channel>.scale*, and if the result is 0.6 the duty cycle will be 60%, and so on. Negative values result in the duty cycle being based on the absolute value, and the direction pin is set to indicate negative.
- *(USC bool input) ppmc.<port>.stepgen.<channel>.enable* - Enables a step pulse generator.
- *(USC real input) ppmc.<port>.stepgen.<channel>.velocity* - Value which determines the step frequency. The value is multiplied by *stepgen.<channel>.scale* , and the result is the frequency in steps per second. Negative values result in the frequency being based on the absolute value, and the direction pin is set to indicate negative.
- *(All bool output) ppmc.<port>.din.<channel>.in* - State of digital input pin, see canonical digital input.

- *(All bool output) ppmc.<port>.din.<channel>.in-not* - Inverted state of digital input pin, see canonical digital input.
- *(All bool input) ppmc.<port>.dout.<channel>.out* - Value to be written to digital output, see canonical digital output.
- *(Option real input) ppmc.<port>.DAC8.<channel>.value* - Value to be written to analog output, range from 0 to 255. This sends 8 output bits to J8, which should have a Spindle DAC board connected to it. 0 corresponds to zero Volts, 255 corresponds to 10 Volts. The polarity of the output can be set for always minus, always plus, or can be controlled by the state of SSR1 (plus when on) and SSR2 (minus when on). You must specify extradac = 0x00 on the HAL command line that loads the PPMC driver to enable this function on the first USC ur UPC board.
- *(Option bool input) ppmc.<port>.dout.<channel>.out* - Value to be written to one of the 8 extra digital output pins on J8. You must specify extradout = 0x00 on the HAL command line that loads the ppmc driver to enable this function on the first USC or UPC board. extradac and extradout are mutually exclusive features as they use the same signal lines for different purposes. These output pins will be enumerated after the standard digital outputs of the board.

6.9.3. Parameter

- *(All real) ppmc.<port>.encoder.<channel>.scale* - The number of counts / user unit (to convert from counts to units).
- *(UPC real) ppmc.<port>.pwm.<channel-range>.freq* - The PWM carrier frequency, in Hz. Applies to a group of four consecutive PWM generators, as indicated by <*channel-range*>. Minimum is 610 Hz, maximum is 500 kHz.
- *(PPMC real) ppmc.<port>.DAC.<channel>.scale* - Sets scale of DAC16 output channel such that an output value equal to the 1/scale value will produce an output of + or - value Volts. So, if the scale parameter is 0.1 and you send a value of 0.5, the output will be 5.0 Volts.
- *(UPC real) ppmc.<port>.pwm.<channel>.scale* - Scaling for PWM generator. If *scale* is X, then the duty cycle will be 100% when the *value* pin is X (or -X).
- *(UPC real) ppmc.<port>.pwm.<channel>.max-dc* - Maximum duty cycle, from 0.0 to 1.0.
- *(UPC real) ppmc.<port>.pwm.<channel>.min-dc* - Minimum duty cycle, from 0.0 to 1.0.
- *(UPC real) ppmc.<port>.pwm.<channel>.duty-cycle* - Actual duty cycle (used mostly for troubleshooting.)
- *(UPC bool) ppmc.<port>.pwm.<channel>.bootstrap* - If true, the PWM generator will generate a short sequence of pulses of both polarities when E-stop goes false, to reset the shutdown latches on some PWM servo drives.
- *(USC uint) ppmc.<port>.stepgen.<channel-range>.setup-time-ns* - Sets minimum time between direction change and step pulse, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs can be specified.
- *(USC uint) ppmc.<port>.stepgen.<channel-range>.pulse-width-ns* - Sets width of step pulses, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs may be specified.
- *(USC uint) ppmc.<port>.stepgen.<channel-range>.pulse-space-min-ns* - Sets minimum time between

pulses, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs can be specified. The maximum step rate is: $\frac{1}{100\text{ns} * (pulsewidth + pulsespacemin)}$

- *(USC real) ppmc.<port>.stepgen.<channel>.scale* - Scaling for step pulse generator. The step frequency in Hz is the absolute value of *velocity* * *scale*.
- *(USC real) ppmc.<port>.stepgen.<channel>.max-vel* - The maximum value for *velocity*. Commands greater than *max-vel* will be clamped. Also applies to negative values. (The absolute value is clamped.)
- *(USC real) ppmc.<port>.stepgen.<channel>.freq* - Actual step pulse frequency in Hz (used mostly for troubleshooting.)
- *(Option real) ppmc.<port>.DAC8.<channel>.scale* - Sets scale of extra DAC output such that an output value equal to scale gives a magnitude of 10.0 V output. (The sign of the output is set by jumpers and/or other digital outputs.)
- *(Option bool) ppmc.<port>.dout.<channel>.invert* - Inverts a digital output, see canonical digital output.
- *(Option bool) ppmc.<port>.dout.<channel>.invert* - Inverts a digital output pin of J8, see canonical digital output.

6.9.4. Funktionen

- *(All funct) ppmc.<Port>.read* - Liest alle Eingänge (digitale Eingänge und Encoderzähler) an einem Port. Diese Lesevorgänge sind in Blöcken von zusammenhängenden Registern organisiert, die in einem Block gelesen werden, um den CPU-Overhead zu minimieren.
- *(All funct) ppmc.<Port>.write* - Schreibt alle Ausgänge (digitale Ausgänge, Stepgen, PWMs) auf einen Port. Diese Schreibvorgänge sind in Blöcken von zusammenhängenden Registern organisiert, die in einem Block geschrieben werden, um den CPU-Overhead zu minimieren.

6.10. Powermax Modbus-Treiber

Dies ist ein in Python geschriebenes nicht-Echtzeit HAL-Programm zur Steuerung von Hypetherm Powermax-Plasmaschneidern unter Verwendung des Modbus-ASCII-Protokolls über RS485.

NOTE Da es sich um ein nicht-Echtzeit-Programm handelt, kann es durch Computerbelastung und Latenzzeiten beeinträchtigt werden. Es ist möglich, die Kommunikation zu verlieren, was durch eine Änderung der Statusausgabe angezeigt wird. Es sollte immer eine Notaus-Schaltung vorhanden sein, um die Stromzufuhr zum Gerät im Notfall verlässlich unterbrechen zu können.

Diese Komponente wird mit dem halcmd-Befehl "loadusr" geladen:

```
loadusr -Wn pmx485 pmx485 /dev/ttyUSB0
```

Dadurch wird die pmx485-Komponente über den Port /dev/ttyUSB0 geladen und gewartet, bis sie bereit

ist.

Es ist erforderlich, den für die Kommunikation zu verwendenden Anschluss zu benennen.

6.10.1. Pins

- **pmx485.mode_set** (real, in) # set cutting mode
- **pmx485.current_set** (real, in) # set cutting current
- **pmx485.pressure_set** (real, in) # set gas pressure
- **pmx485.enable** (bool, in) # enable the component
- **pmx485.mode** (real, out) # cut mode feedback
- **pmx485.current** (real, out) # cutting current feedback
- **pmx485.pressure** (real, out) # gas pressure feedback
- **pmx485.fault** (real, out) # powermax fault code
- **pmx485.status** (bool, out) # connection status
- **pmx485.current_min** (real, out) # minimum allowed current
- **pmx485.current_max** (real, out) # maximum allowed current
- **pmx485.pressure_min** (real, out) # minimum allowed gas pressure
- **pmx485.pressure_max** (real, out) # maximum allowed gas pressure
- **pmx485.arcTime** (real, out) # arc on time feedback

6.10.2. Beschreibung

Um mit einem Powermax zu kommunizieren, muss die Komponente zunächst über den **enable**-Pin aktiviert werden und kann dann eine Anfrage an den Powermax stellen, indem sie einen gültigen String an die folgenden Pins schreibt:

- **mode_set**
- **current_set**
- **pressure_set**

NOTE	A **pressure_set** value of zero is valid, the Powermax will then calculate the required pressure internally.

Die Kommunikation kann über das Powermax-Display oder den **Status**-Pin validiert werden. Im Fernsteuerungsmodus können der Modus, der Strom und der Druck nach Bedarf geändert werden.

Um die Kommunikation zu beenden, führen Sie einen der folgenden Schritte aus:

- Set all set pins to zero: **mode_set**, **current_set**, and **pressure_set**.
- Trennen Sie das Powermax-Netzteil für etwa 30 Sekunden von der Stromquelle. Wenn Sie das System wieder einschalten, befindet es sich nicht mehr im Remote-Modus.

6.10.3. Referenz:

- Hypertherm Anwendungshinweis #807220
  "Powermax45 XP/65/85/105/125® Serielles Kommunikationsprotokoll"

6.11. Shuttle

6.11.1. Beschreibung

Shuttle ist eine Nicht-Echtzeit-HAL-Komponente, die Schnittstellen Contour Design's ShuttleXpress, ShuttlePRO, und ShuttlePRO2 Geräte mit LinuxCNC's HAL.

Wenn der Treiber ohne Befehlszeilenargumente gestartet wird, sucht er in allen /dev/hidraw*-Gerätedateien nach Shuttle-Geräten und verwendet alle gefundenen Geräte. Wenn er mit Befehlszeilenargumenten gestartet wird, prüft er nur die angegebenen Geräte.

Das ShuttleXpress verfügt über fünf Taster, ein Jogwheel mit 10 Zählern/Umdrehung und ein federbelastetes Außenrad mit 15 Positionen, das beim Loslassen in die Mitte zurückkehrt.

Das ShuttlePRO verfügt über 13 Tasten, ein Jogwheel mit 10 Zählern/Umdrehung und ein federbelastetes Außenrad mit 15 Positionen, das beim Loslassen in die Mitte zurückkehrt.

Das ShuttlePRO2 verfügt über 15 Drucktasten, ein Jogwheel mit 10 Zählern/Umdrehung und ein federbelastetes Außenrad mit 15 Positionen, das beim Loslassen in die Mitte zurückkehrt.

Die Shuttle-Geräte verfügen über einen internen 8-Bit-Zähler für die aktuelle Jog-Wheel-Position. Der Shuttle-Treiber kann diesen Wert nicht kennen, bis das Shuttle-Gerät sein erstes Ereignis sendet. Wenn das erste Ereignis beim Treiber eingeht, verwendet der Treiber die vom Gerät gemeldete Jogwheel-Position, um den Zähler auf 0 zu initialisieren.

WARNING

Das heißt, wenn das erste Ereignis durch eine Jogwheel-Bewegung erzeugt wird, geht diese erste Bewegung verloren.

Jede Benutzerinteraktion mit dem Shuttle-Gerät erzeugt ein Ereignis, das den Fahrer über die Position des Jogwheels informiert. Wenn Sie also (zum Beispiel) beim Start eine der Tasten drücken, funktioniert das Jog-Wheel einwandfrei und bemerkt den ersten Klick.

6.11.2. Einrichtung

Der Shuttle-Treiber benötigt Leserechte für die Gerätedateien /dev/hidraw*. Dies kann durch Hinzufügen einer Datei /etc/udev/rules.d/99-shuttle.rules mit dem folgenden Inhalt erreicht werden:

```
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="0b33", ATTRS{idProduct}=="0020", MODE="0444"
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="05f3", ATTRS{idProduct}=="0240", MODE="0444"
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="0b33", ATTRS{idProduct}=="0030", MODE="0444"
```

Das LinuxCNC Debian-Paket installiert eine entsprechende udev-Datei automatisch, aber wenn Sie LinuxCNC aus dem Quellcode bauen und nicht das Debian-Paket verwenden, müssen Sie diese Datei von Hand installieren. Wenn Sie die Datei von Hand installieren, müssen Sie udev anweisen, seine Regeldateien neu zu laden, indem Sie `udevadm control --reload-rules` ausführen.

6.11.3. Pins

Allen HAL-Pin-Namen wird das Präfix "shuttle" vorangestellt, gefolgt vom Index des Geräts (die Reihenfolge, in der sie vom Treiber gefunden wurden), z. B. `shuttle.0` oder `shuttle.2`.

<*Prefix*>`.button-<ButtonNumber>` (bool out)

Diese Pins sind wahr (1) wenn die Schaltfläche gedrückt wird.

<*Prefix*>`.button-<ButtonNumber>-not` (bool out)

Diese Pins haben den umgekehrten Zustand des Buttons, also sind sie True (1) wenn der Button nicht gedrückt wird.

<*Prefix*>`.counts` (sint out)

Kumulierte Zählungen des Jogwheels (das innere Rad).

<*Prefix*>`.spring-wheel-s32` (sint out)

Die aktuelle Auslenkung des Federrads (des äußeren Rads). Im Ruhezustand ist er 0 und reicht von -7 am äußersten Ende gegen den Uhrzeigersinn bis +7 am äußersten Ende im Uhrzeigersinn.

<*Prefix*>`.spring-wheel-f` (real out)

Die aktuelle Auslenkung des Federrads (des äußeren Rads). Sie beträgt 0,0 im Ruhezustand, -1,0 im extremen Gegenuhrzeigersinn und +1,0 im extremen Uhrzeigersinn. Die Shuttle-Geräte melden die Position des Federrads als Ganzzahl von -7 bis +7, so dass dieser Pin nur 15 diskrete Werte in seinem Bereich meldet.

6.12. VFS11 VFD-Treiber

Dies ist ein nicht-Echtzeit HAL-Programm zur Steuerung der S11-Serie von VFDs von Toshiba.

vfs11_vfd unterstützt serielle und TCP-Verbindungen. Serielle Verbindungen können RS232 oder RS485 sein. RS485 wird im Voll- und Halbduplex-Modus unterstützt. TCP-Verbindungen können passiv (Warten auf eine eingehende Verbindung) oder aktiv (ausgehende Verbindungen) sein, was für die Verbindung mit TCP-basierten Geräten oder über einen Terminal-Server nützlich sein kann.

Unabhängig von der Verbindungsart arbeitet vfs11_vfd als Modbus-Master.

Diese Komponente wird mit dem halcmd-Befehl "loadusr" geladen:

```
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd
```

Der obige Befehl lautet: loadusr, warten bis named geladen ist, Komponente vfs11_vfd, named spindle-vfd

6.12.1. Kommandozeilen-Optionen

vfs11_vfd wird hauptsächlich durch INI-Datei-Optionen konfiguriert. Die Kommandozeilenoptionen sind:

- *-n oder --name <halname>* : den Namen der HAL-Komponente festlegen
- *-I oder --ini <inifilename>* : übernimmt die Konfiguration aus der angegebenen INI-Datei. Standardmäßig wird die Umgebungsvariable INI_FILE_NAME verwendet.
- *-S oder --section <section name>* : übernimmt die Konfiguration aus dem Abschnitt mit diesem Namen in der INI-Datei. Der Standardwert ist *VFS11*.
- *-d oder --debug* aktiviert Debug-Meldungen in der Konsolenausgabe.
- *-m oder --modbus-debug* aktiviert Modbus-Meldungen auf der Konsolenausgabe
- *-r oder --report-device* Geräteeigenschaften beim Starten auf der Konsole melden

Das Debugging kann durch Senden eines USR1-Signals an den vfs11_vfd-Prozess umgeschaltet werden. Modbus-Debugging kann durch Senden eines USR2-Signals an den vfs11_vfd-Prozess umgeschaltet werden (Beispiel: `kill -USR1 \`pidof vfs11_vfd\``).

NOTE Bei seriellen Konfigurationsfehlern kann das Einschalten von verbose zu einer Flut von Timeout-Fehlern führen.

6.12.2. Pins

Dabei steht <n> für `vfs11_vfd` oder den beim Laden mit der Option -n angegebenen Namen.

- *<n>.acceleration-pattern* (bool, in) when true, set acceleration and deceleration times as defined in registers F500 and F501 respectively. Wird in PID Schleifen verwendet, um kürzere Rampenzeiten zu wählen und so Schwingungen zu vermeiden.
- *<n>.alarm-code* (sint, out) non-zero if drive is in alarmed state. Bitmap describing alarm information (see register FC91 description). Use err-reset (see below) to clear the alarm.
- *<n>.at-speed* (bool, out) when drive is at commanded speed (see speed-tolerance below)
- *<n>.current-load-percentage* (real, out) reported from the VFD
- *<n>.dc-brake* (bool, in) engage the DC brake. Also turns off spindle-on.
- *<n>.enable* (bool, in) enable the VFD. If false, all operating parameters are still read but control is released and panel control is enabled (subject to VFD setup).
- *<n>.err-reset* (bool, in) reset errors (alarms a.k.a Trip and e-stop status). Resetting the VFD may cause a 2-second delay until it's rebooted and Modbus is up again.
- *<n>.estop* (bool, in) put the VFD into emergency-stopped status. No operation possible until cleared with err-reset or powercycling.
- *<n>.frequency-command* (real, out) current target frequency in Hz as set through speed-command (which is in RPM), from the VFD
- *<n>.frequency-out* (real, out) current output frequency of the VFD
- *<n>.inverter-load-percentage* (real, out) current load report from VFD

- *<n>.is-e-stopped* (bool, out) the VFD is in emergency stop status (blinking "E" on panel). Use err-reset to reboot the VFD and clear the e- stop status.
- *<n>.is-stopped* (bool, out) true when the VFD reports 0 Hz output
- *<n>.max-rpm* (real, R) actual RPM limit based on maximum frequency the VFD may generate, and the motors nameplate values. For instance, if nameplate-HZ is 50, and nameplate-RPM_ is 1410, but the VFD may generate up to 80 Hz, then max-rpm would read as 2256 (80*1410/50). The frequency limit is read from the VFD at startup. To increase the upper frequency limit, the UL and FH parameters must be changed on the panel. See the VF-S11 manual for instructions how to set the maximum frequency.
- *<n>.modbus-ok* (bool, out) true when the Modbus session is successfully established and the last 10 transactions returned without error.
- *<n>.motor-RPM* (real, out) estimated current RPM value, from the VFD
- *<n>.output-current-percentage* (real, out) from the VFD
- *<n>.output-voltage-percentage* (real, out) from the VFD
- *<n>.frequency-limit* (real, out) upper limit read from VFD setup.
- *<n>.speed-command* (real, in) speed sent to VFD in RPM. It is an error to send a speed faster than the Motor Max RPM as set in the VFD
- *<n>.spindle-fwd* (bool, in) 1 for FWD and 0 for REV, sent to VFD
- *<n>.spindle-on* (bool, in) 1 for ON and 0 for OFF sent to VFD, only on when running
- *<n>.spindle-rev* (bool, in) 1 for ON and 0 for OFF, only on when running
- *<n>.jog-mode* (bool, in) 1 for ON and 0 for OFF, enables the VF-S11 *jog mode*. Speed control is disabled, and the output frequency is determined by register F262 (preset to 5 Hz). This might be useful for spindle orientation. In normal mode, the VFD shuts off if the frequency drops below 12 Hz.
- *<n>.status* (sint, out) Drive Status of the VFD (see the TOSVERT VF-S11 Communications Function Instruction Manual, register FD01). A bitmap.
- *<n>.trip-code* (sint, out) trip code if VF-S11 is in tripped state.
- *<n>.error-count* (sint, out) number of Modbus transactions which returned an error
- *<n>.max-speed* (bool, in) ignore the loop-time parameter and run Modbus at maximum speed, at the expense of higher CPU usage. Suggested use during spindle positioning.

6.12.3. Parameter

Dabei steht <n> für `vfs11_vfd` oder den beim Laden mit der Option -n angegebenen Namen.

- *<n>.loop-time* (real, RW) how often the Modbus is polled (default interval 0.1 seconds)
- *<n>.nameplate-HZ* (real, RW) Nameplate Hz of motor (default 50). Used to calculate target frequency (together with nameplate-RPM) for a target RPM value as given by speed-command.
- *<n>.nameplate-RPM* (real, RW) Nameplate RPM of motor (default 1410)
- *<n>.rpm-limit* (real, RW) do-not-exceed soft limit for motor RPM (defaults to nameplate-RPM).

- *<n>.tolerance* (real, RW) speed tolerance (default 0.01) for determining whether spindle is at speed (0.01 meaning: Output frequency is within 1% of target frequency)

6.12.4. INI-Datei-Konfiguration

Hier werden alle Optionen aufgelistet, die vfs11_vfd versteht. Typische Einstellungen für RS-232, RS-485 und TCP finden Sie in *src/hal/user_comps/vfs11_vfd/*.ini*.

```
[VFS11]
# serielle Verbindung
TYPE=rtu

# serielle Schnittstelle
DEVICE=/dev/ttyS0

# TCP-Server - Warten Sie auf eingehende Verbindung
TYP=tcpserver

# tcp portnumber für TYPE=tcpserver oder tcpclient
PORT=1502

# TCP-Client - aktive ausgehende Verbindung
TYPE=tcpclient

# Ziel, zu dem eine Verbindung aufgebaut werden soll if TYPE=tcpclient
TCPDEST=192.168.1.1

---------- nur sinnvoll, wenn TYPE=rtu -------
# serial device detail
# 5 6 7 8
BITS= 5

# even odd none
# (engl. für gerade ungerade keine)
PARITY=none

# 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
BAUD=19200

# 1 2
STOPBITS=1

#rs232 rs485
SERIAL_MODE=rs485

# up down none (engl. für auf ab keine)
# diese Funktion funktioniert möglicherweise nicht mit einem Ubuntu-Paket
# libmodbus5/libmodbus-dev-Paket und erzeugt eine Warnung.
# Die Ausführung wird fortgesetzt, als ob RTS_MODE=up angegeben wäre.
RTS_MODE=up
#---------------------

# Modbus-Timer in Sekunden
# inter-character timer
BYTE_TIMEOUT=0.5
```

```
# packet timer
RESPONSE_TIMEOUT=0.5

# Ziel (engl. target)-Modbus-ID
TARGET=1

# bei E/A-Fehlern wird versucht, die Verbindung nach einer Verzögerung von
# RECONNECT_DELAY Sekunden wiederherzustellen
RECONNECT_DELAY=1

# Verschiedene weitere Parameter
DEBUG=10
MODBUS_DEBUG=0
POLLCYCLES=10
```

6.12.5. HAL-Beispiel

```
#
# Beispiel für die Verwendung des VF-S11 VFD-Treibers
#
#
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd

# Verbinden der Spindelrichtungs-Pins mit dem VFD
net vfs11-fwd spindle-vfd.spindle-fwd <= spindle.0.forward
net vfs11-rev spindle-vfd.spindle-rev <= spindle.0.reverse

# Verbinden des Pins "Spindel ein" mit dem VF-S11
net vfs11-run spindle-vfd.spindle-on <= spindle.0.on

# Verbinden des VF-S11-auf-Geschwindigkeit-Pins
net vfs11-at-speed spindle.0.at-speed <= spindle-vfd.at-speed

# Verbinden der Spindeldrehzahl-Pins
net vfs11-RPM spindle-vfd.speed-command <= spindle.0.speed-out

# Anschluss der Gleichstrombremse an VF-S11
# Da diese bei ausgeschalteter Spindel Strom zieht, sollte der DC-Brems-Pin
# besser von einem Monoflop angesteuert werden, das bei der fallenden Flanke von "Spindel
ein" auslöst
#net vfs11-spindle-brake spindle.N.brake => spindle-vfd.dc-brake

# um den VFS11-Jog-Modus für die Spindelausrichtung zu verwenden
# siehe orient.9 und motion.9
net spindle-orient spindle.0.orient spindle-vfd.max-speed spindle-vfd.jog-mode

# hat Vorrang vor dem Bedienfeld
setp spindel-vfd.enable 1
```

6.12.6. Bedienung des Panels

Der vfs11_vfd-Treiber hat Vorrang vor der Panel-Steuerung, solange er aktiviert ist (siehe *enable* Pin), wodurch das Panel effektiv deaktiviert wird. Durch Löschen des *enable* Pins wird das Panel wieder

aktiviert. Pins und Parameter können weiterhin eingestellt werden, doch werden diese nicht in den VFD geschrieben, solange der *enable* Pin nicht gesetzt ist. Betriebsparameter werden weiterhin gelesen, während die Bussteuerung deaktiviert ist. Durch kontrolliertes Beenden des vfs11_vfd-Treibers wird der VFD vom Bus getrennt und die Kontrolle über das Panel wiederhergestellt.

Weitere Informationen finden Sie im LinuxCNC Integrators Manual. Eine detaillierte Registerbeschreibung der Toshiba VFDs finden Sie im "TOSVERT VF-S11 Communications Function Instruction Manual" (Toshiba Dokumentnummer E6581222) und im "TOSVERT VF-S11 Instruction manual" (Toshiba Dokumentnummer E6581158).

6.12.7. Reaktion auf Fehler (engl. error recovery)

Der `vfs11_vfd` erholt sich von E/A-Fehlern wie folgt: Zunächst werden alle HAL-Pins auf Standardwerte gesetzt, und der Treiber schläft für `RECONNECT_DELAY` Sekunden (Standard 1 Sekunde).

- Serieller Modus (`TYPE=rtu`): Bei einem Fehler wird die serielle Schnittstelle geschlossen und erneut geöffnet.
- TCP-Server-Modus (`TYPE=tcpserver`): Wenn die TCP-Verbindung unterbrochen wird, schaltet der Treiber wieder auf die Suche nach eingehenden Verbindungen.
- TCP-Client-Modus (`TYPE=tcpclient`): Bei Verlust der TCP-Verbindung stellt der Treiber die Verbindung zu *TCPDEST:PORTNO* wieder her.

6.12.8. Konfigurieren des VFS11 VFD für die Modbus-Nutzung

Anschließen der seriellen Schnittstelle

Der VF-S11 verfügt über eine RJ-45-Buchse für die serielle Kommunikation. Leider verfügt es nicht über einen Standard-RS-232-Stecker und logische Pegel. Der von Toshiba empfohlene Weg ist: Schließen Sie die USB001Z USB-zu-Seriell-Konvertierungseinheit an das Laufwerk an, und verbinden Sie den USB-Anschluss mit dem PC. Eine billigere Alternative ist eine selbstgebaute Schnittstelle (Hinweise vom Toshiba-Support, Schaltplan).

Hinweis: Der 24-V-Ausgang des VFD hat keinen Kurzschlussschutz.

Die Werkseinstellungen für die serielle Schnittstelle sind 9600/8/1/gerade, das Protokoll ist standardmäßig das proprietäre Toshiba Inverter Protocol".

Modbus-Einrichtung

Mehrere Parameter müssen eingestellt werden, bevor der VF-S11 mit diesem Modul kommunizieren kann. Dies kann entweder manuell über das Bedienfeld oder über die serielle Verbindung erfolgen - Toshiba liefert eine Windows-Anwendung namens *PCM001Z*, die Parameter im VFD lesen/einstellen kann. Hinweis - PCM001Z spricht nur das Toshiba Wechselrichterprotokoll. Der letzte Parameter, den Sie ändern möchten, ist das Protokoll - stellen Sie es von Toshiba Inverter Protocol auf Modbus um; danach ist die Windows-Anwendung nutzlos.

Um die obere Frequenzgrenze zu erhöhen, müssen die Parameter UL und FH auf dem Panel geändert

werden. Ich habe sie von 50 auf 80 erhöht.

Siehe dump-params.mio für eine Beschreibung der nicht standardmäßigen VF-S11-Parameter meiner Einrichtung. Diese Datei ist für das modio Modbus interactive utility.

6.12.9. Hinweis zur Programmierung

Der Treiber vfs11_vfd verwendet die Bibliothek libmodbus Version 3, die aktueller ist als der in `gs2_vfd` verwendete Code der Version 2.

Die Ubuntu-Pakete `libmodbus5` und `libmodbus-dev` sind erst ab Ubuntu 12 (*Precise Pengolin*) verfügbar. Außerdem fehlt diesen Paketen die Unterstützung für die MODBUS_RTS_MODE_*-Flags. Daher kann das Erstellen von vfs11_vfd mit dieser Bibliothek eine Warnung erzeugen, wenn RTS_MODE= in der INI-Datei angegeben ist.

Um die volle Funktionalität auf Ubuntu Lucid und Precise zu nutzen:

- Entfernen Sie die libmodbus-Pakete: `sudo apt-get remove libmodbus5 libmodbus-dev`
- erstellen und installieren Sie libmodbus Version 3 aus den Quellen, wie unter hier beschrieben.

Libmodbus kann nicht auf Ubuntu Hardy gebaut werden, daher ist vfs11_vfd auf Hardy nicht verfügbar.

[1] In Europa kann das Äquivalent unter dem Markennamen Omron gefunden werden.

Chapter 7. Hardware-Beispiele

7.1. PCI-Parallelport

Wenn Sie eine zweite parallele Schnittstelle zu Ihrem PCI-Bus hinzufügen, müssen Sie die Adresse herausfinden, bevor Sie sie mit LinuxCNC verwenden können.

Um die Adresse Ihrer Parallelportkarte zu ermitteln, öffnen Sie ein Terminalfenster und geben Sie ein

```
lspci -v
```

Sie werden etwas Ähnliches sehen, wie auch Informationen über alles andere auf dem PCI-Bus:

```
0000:00:10.0 Communication controller: \
        NetMos Technology PCI 1 port parallel adapter (rev 01)
        Subsystem: LSI Logic / Symbios Logic: Unknown device 0010
        Flags: medium devsel, IRQ 11
        I/O ports at a800 [size=8]
        I/O ports at ac00 [size=8]
        I/O ports at b000 [size=8]
        I/O ports at b400 [size=8]
        I/O ports at b800 [size=8]
        I/O ports at bc00 [size=16]
```

In meinem Fall war die Adresse die erste, also änderte ich meine .hal-Datei von

```
loadrt hal_parport cfg=0x378
```

zu

```
loadrt hal_parport cfg="0x378 0xa800 in"
```

(Beachten Sie die Anführungszeichen, in denen die Adressen stehen.)

und fügte dann die folgenden Zeilen hinzu, damit der Parport gelesen und geschrieben werden kann:

```
addf parport.1.read base-thread
addf parport.1.write base-thread
```

Nachdem Sie die obigen Schritte durchgeführt haben, starten Sie mit dieser Konfiguration und überprüfen Sie, ob im Fenster Maschine/HAL-Konfiguration angezeigt wird, dass die parallele Schnittstelle geladen wurde.

7.2. Spindelsteuerung

LinuxCNC kann bis zu 8 Spindeln steuern. Die Anzahl wird in der INI-Datei eingestellt. Die Beispiele unten beziehen sich alle auf eine Einspindelkonfiguration mit Spindelsteuerungspins mit Namen wie

spindle.0... Im Falle einer Mehrspindelmaschine ist alles, was sich ändert, dass zusätzliche Pins mit Namen wie *spindle.6...* existieren.

7.2.1. 0-10 Volt Spindeldrehzahl

Wenn Ihre Spindeldrehzahl durch ein analoges Signal gesteuert wird (z. B. durch einen VFD mit einem 0 V bis 10 V-Signal) und Sie eine DAC-Karte wie die m5i20 zur Ausgabe des Steuersignals verwenden:

Zunächst müssen Sie die Skala von Spindeldrehzahl zu Steuersignal, das ist die anliegende Spannung, ermitteln. In diesem Beispiel entspricht die Höchstgeschwindigkeit der Spindel von 5000 U/min 10 Volt.

$$\frac{10\,Volts}{5000\,RPM} = \frac{0.002\,Volts}{1\,RPM}$$

Wir müssen der HAL-Datei eine Skalierungskomponente hinzufügen, um die *spindle.N.speed-out* auf die vom VFD benötigten Werte 0 bis 10 zu skalieren, wenn Ihre DAC-Karte keine Skalierung vornimmt.

```
loadrt scale count=1
addf scale.0 servo-thread
setp scale.0.gain 0.002
net spindle-speed-scale spindle.0.speed-out => scale.0.in
net spindle-speed-DAC scale.0.out => <Ihr DAC Pin-Name>
```

7.2.2. PWM Spindeldrehzahl

Wenn Ihre Spindel durch ein PWM-Signal gesteuert werden kann, verwenden Sie die Komponente „pwmgen“, um das Signal zu erzeugen:

```
loadrt pwmgen output_type=0
addf pwmgen.update servo-thread
addf pwmgen.make-pulses base-thread
net spindle-speed-cmd spindle.0.speed-out => pwmgen.0.value
net spindle-on spindle.0.on => pwmgen.0.enable
net spindle-pwm pwmgen.0.pwm => parport.0.pin-09-out
# Set the spindle's top speed in RPM
setp pwmgen.0.scale 1800
```

Dabei wird davon ausgegangen, dass die Spindelsteuerung einfach auf PWM reagiert: 0 % PWM ergibt 0 U/min, 10 % PWM ergibt 180 U/min, usw. Wenn eine Mindest-PWM erforderlich ist, um die Spindel zum Drehen zu bringen, folgen Sie dem Beispiel in der Beispielkonfiguration der nist-lathe und verwenden Sie eine Skalierungskomponente.

7.2.3. Spindel Aktivierung

Wenn Sie ein Spindelaktivierungssignal benötigen, verknüpfen Sie Ihren Ausgangspin mit *spindle.0.on*. Um diese Pins mit einem Parallelport-Pin zu verknüpfen, fügen Sie etwas wie das Folgende in Ihre .hal-Datei ein, wobei Sie darauf achten, dass Sie den Pin auswählen, der mit Ihrem Steuergerät verbunden ist.

```
net spindle-enable spindle.0.on => parport.0.pin-14-out
```

7.2.4. Spindel Richtung

Wenn Sie die Kontrolle über die Drehrichtung Ihrer Spindel haben, dann werden die HAL-Pins *spindle.N.forward* und *spindle.N.reverse* durch die G-Codes M3 und M4 gesteuert. Die Spindeldrehzahl S*n* muss auf einen positiven Wert ungleich Null eingestellt werden, damit M3/M4 die Spindelbewegung einschalten kann.

Um diese Pins mit einem Parallelport-Pin zu verknüpfen, fügen Sie etwas wie das Folgende in Ihre .hal-Datei ein und stellen sicher, dass Sie den Pin auswählen, der mit Ihrem Steuergerät verbunden ist.

```
net spindle-fwd spindle.0.forward => parport.0.pin-16-out
net spindle-rev spindle.0.reverse => parport.0.pin-17-out
```

7.2.5. Spindel Soft Start

Wenn Sie Ihren Spindeldrehzahl-Befehl rampenförmig erhöhen müssen und Ihre Steuerung nicht über diese Funktion verfügt, kann dies in HAL erfolgen. Grundsätzlich müssen Sie die Ausgabe von *spindle.N.speed-out* zu entführen und führen Sie es durch eine *limit2*-Komponente mit der Skala eingestellt, so dass es die Drehzahl von *spindle.N.speed-out* zu Ihrem Gerät, das die Drehzahl empfängt Rampe wird. Der zweite Teil ist es, LinuxCNC wissen, wenn die Spindel bei der Geschwindigkeit, so dass die Bewegung beginnen kann.

In dem 0-10-Volt-Beispiel wird hierzu die Zeile

```
net spindle-speed-scale spindle.0.speed-out => scale.0.in
```

wie im folgenden Beispiel geändert:

Einführung in die HAL-Komponenten limit2 und near

Für den Fall, dass Sie sie noch nicht kennen, hier eine kurze Einführung in die beiden HAL-Komponenten, die im folgenden Beispiel verwendet werden.

- Ein *limit2* ist eine HAL-Komponente (Fließkomma), die einen Eingangswert akzeptiert und einen Ausgang liefert, der auf einen Max/Min-Bereich begrenzt wurde und außerdem eine bestimmte Änderungsrate nicht überschreiten darf.
- *near* ist eine HAL-Komponente (Gleitkomma) mit einem binären Ausgang, der angibt, ob zwei Eingaben ungefähr gleich sind.

Weitere Informationen finden Sie in der Dokumentation zu den HAL-Komponenten oder in den Manpages, sagen Sie einfach *man limit2* oder *man near* in einem Terminal.

```
# Legen Sie die Instanzen der Echtzeit-Module limit2 und near mit Namen an, damit die
nachfolgenden Verbindungen einfacher zu verfolgen sind
loadrt limit2 names=spindel-ramp
loadrt near names=spindel-at-speed
```

```
# die Funktionen zu einem Thread hinzufügen
addf spindle-ramp servo-thread
addf spindle-at-speed servo-thread

# den Parameter für die maximale Änderungsrate einstellen
# (maximale Spindelbeschleunigung/-verzögerung in Einheiten pro Sekunde)
setp spindle-ramp.maxv 60

# Die Spindeldrehzahl an die Spindelrampe umlenken
net spindle-cmd <= spindle.0.speed-out => spindle-ramp.in

# die Ausgabe der Spindelrampe wird an die Sklaierung gesendet
net spindle-ramped <= spindle-ramp.out => scale.0.in

# um zu wissen, wann die Bewegung beginnen soll, senden wir die Nahkomponente
# (namens spindle-at-speed) an die Spindeldrehzahl aus
# dem Signal spindle-cmd und der tatsächlichen Spindeldrehzahl
# vorausgesetzt, Ihre Spindel kann mit der maxv-Einstellung beschleunigen.
net spindle-cmd => spindle-at-speed.in1
net spindle-ramped => spindle-at-speed.in2

# die Ausgabe von spindle-at-speed wird an spindle.0.at-speed gesendet
# und wenn dies wahr ist, beginnt die Bewegung
net spindle-ready <= spindle-at-speed.out => spindle.0.at-speed
```

7.2.6. Spindel-Feedback

Spindelsynchronisierte Bewegung

Spindel-Feedback wird von LinuxCNC benötigt, um alle Spindel koordinierte Bewegungen wie Gewindeschneiden und konstante Oberflächengeschwindigkeit (engl. constant surface speed, kurz CSS) durchzuführen. LinuxCNC kann synchronisierte Bewegung und CSS mit bis zu 8 Spindeln durchführen. Welche Spindeln verwendet werden, wird von G-Code gesteuert. CSS ist mit mehreren Spindeln gleichzeitig möglich.

Der StepConf Wizard kann die Verbindungen für eine Einspindelkonfiguration für Sie durchführen, wenn Sie Encoder Phase A und Encoder Index als Eingänge auswählen.

Hardware-Annahmen für dieses Beispiel:

- An der Spindel ist ein Drehgeber angeschlossen, der auf der Phase A 100 Impulse pro Umdrehung ausgibt.
- Die A-Phase des Encoders wird an den Pin 10 des Parallelports angeschlossen.
- Der Indeximpuls des Encoders wird an den Parallelport Pin 11 angeschlossen.

Grundlegende Schritte, um die Komponenten hinzuzufügen und zu konfigurieren: [1] [2] [3]

```
# Fügen Sie den Encoder zu HAL hinzu und verbinden Sie ihn mit Threads.
loadrt encoder num_chan=4
addf encoder.update-counters base-thread
```

```
addf encoder.capture-position servo-thread

# Den HAL-Geber auf 100 Impulse pro Umdrehung einstellen.
setp encoder.3.position-scale 100

# Stellen Sie den HAL-Encoder auf einfache Zählung ohne Quadratur nur auf A ein.
setp encoder.3.counter-mode true

# Verbinden Sie die HAL-Geberausgänge mit LinuxCNC.
net spindle-position encoder.3.position => spindle.0.revs
net spindle-velocity encoder.3.velocity => spindle.0.speed-in
net spindle-index-enable encoder.3.index-enable <=> spindle.0.index-enable

# Verbinden Sie die HAL-Encodereingänge mit dem realen Encoder.
net spindle-phase-a encoder.3.phase-A <= parport.0.pin-10-in
net spindle-phase-b encoder.3.phase-B
net spindle-index encoder.3.phase-Z <= parport.0.pin-11-in
```

Spindel bei Drehzahl

Damit LinuxCNC zu warten, bis die Spindel bei der Geschwindigkeit vor der Ausführung einer Reihe von Bewegungen, die spindle.*N*.at-Geschwindigkeit muss wahr in dem Moment die Spindel ist bei der befohlenen Geschwindigkeit zu drehen. Um dies zu erreichen, benötigen Sie ein Spindel-Feedback von einem Encoder. Da die Rückmeldung und die befohlene Drehzahl in der Regel nicht "genau" übereinstimmen, sollten Sie die Komponente "nahe" verwenden, um festzustellen, ob die beiden Zahlen nahe genug beieinander liegen.

Die benötigten Verbindungen sind vom Spindeldrehzahl-Sollwertsignal zu near.n.in1 und von der Spindeldrehzahl vom Encoder zu near.n.in2. Dann wird der near.n.out mit spindle.*N*.at-speed verbunden. Die near.n.scale muss so eingestellt werden, dass sie angibt, wie nahe die beiden Zahlen beieinander liegen müssen, bevor der Ausgang aktiviert wird. Je nach Ihrer Einrichtung müssen Sie die Skala möglicherweise an Ihre Hardware anpassen.

Die folgenden Angaben sind typisch für die Ergänzungen, die in Ihrer HAL-Datei erforderlich sind, um Spindle At Speed zu aktivieren. Wenn Sie in Ihrer HAL-Datei bereits "near" haben, erhöhen Sie die Anzahl und passen Sie den Code entsprechend an. Vergewissern Sie sich, dass die Signalnamen in Ihrer HAL-Datei identisch sind.

```
# Eine near-Komponente laden und an einen Thread anhängen.
loadrt near
addf near.0 servo-thread

# Einen Eingang mit der befohlenen Spindeldrehzahl verbinden.
net spindle-cmd => nahe.0.in1

# Einen Eingang mit der vom Encoder gemessenen Spindelgeschwindigkeit verbinden.
net spindle-velocity => near.0.in2

# Ausgang mit dem Eingang "Spindel-at-speed" verbinden.
net spindle-at-speed spindle.0.at-speed <= near.0.out

# Die Spindeldrehzahleingänge werden so eingestellt, dass sie innerhalb von 1%
```

```
übereinstimmen.
setp near.0.scale 1.01
```

7.3. MPG Handgeräte

In diesem Beispiel wird erklärt, wie die heute auf dem Markt befindlichen MPG-Hängegeräte angeschlossen werden können. In diesem Beispiel wird ein MPG3-Pendant und eine C22-Pendant-Schnittstellenkarte von CNC4PC verwendet, die an einen zweiten parallelen Port angeschlossen ist, der in den PCI-Steckplatz gesteckt wird. Dieses Beispiel bietet Ihnen 3 Achsen mit 3 Schrittweiten von 0,1, 0,01, 0,001

Fügen Sie in Ihrer Datei custom.hal oder jog.hal Folgendes hinzu, wobei Sie sicherstellen müssen, dass Sie nicht bereits mux4 oder einen Encoder verwenden. Falls doch, erhöhen Sie einfach die Anzahl und ändern Sie die Referenznummern. Weitere Informationen über mux4 und encoder finden Sie im HAL-Handbuch oder in der Manpage.

Weitere Informationen zum Hinzufügen einer HAL-Datei finden Sie im Abschnitt INI HAL der Dokumentation. Für jedes Gelenk und alle Koordinatenbuchstaben sind Jog-Management-Pins vorgesehen. In diesem Beispiel werden die Achsen-Jogging-Pins für das Jogging im Weltmodus verwendet. Bei Maschinen mit nicht-identischen Kinematiken müssen möglicherweise zusätzliche Verbindungen für das Jogging im Gelenkmodus verwendet werden.

jog.hal

```
# Jog Pendant
loadrt encoder num_chan=1
loadrt mux4 count=1
addf encoder.capture-position servo-thread
addf encoder.update-counters base-thread
addf mux4.0 servo-thread

# Wenn Ihr MPG ein Quadratursignal pro Klick ausgibt, setzen Sie x4 auf 1.
# Wenn Ihr MPG 1 Impuls pro Klick ausgibt, setzen Sie x4 auf 0
setp encoder.0.x4-mode 0

# Für den Geschwindigkeitsmodus, auf 1 setzen.
# Im Geschwindigkeitsmodus hält die Achse an, wenn der Drehknopf gestoppt wird.
# auch wenn das bedeutet, dass die befohlene Bewegung nicht abgeschlossen ist,
# Für den Positionsmodus (die Voreinstellung), setzen Sie auf 0.
# Im Positionsmodus bewegt sich die Achse bei jeder Zählung genau jog-scale viele
# Einheiten für jede Zählung, unabhängig davon, wie lange das dauern könnte,
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0

# Hier wird die Skala festgelegt, die auf der Grundlage des Eingangs am mux4 verwendet
wird.
setp mux4.0.in0 0.1
setp mux4.0.in1 0.01
setp mux4.0.in2 0.001

# Die Eingänge der Komponente mux4
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
```

```
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

# Die Ausgabe des mux4 wird an jede Achsen-Jog-Skala gesendet
net mpg-scale <= mux4.0.out
net mpg-scale => axis.x.jog-scale
net mpg-scale => axis.y.jog-scale
net mpg-scale => axis.z.jog-scale

# Die MPG Eingänge
net mpg-a encoder.0.phase-A <= parport.1.pin-02-in
net mpg-b encoder.0.phase-B <= parport.1.pin-03-in

# The Achsen-Auswahl Eingänge
net mpg-x axis.x.jog-enable <= parport.1.pin-04-in
net mpg-y axis.y.jog-enable <= parport.1.pin-05-in
net mpg-z axis.z.jog-enable <= parport.1.pin-06-in

# Der Encoder-Ausgang zählt für die Achse. Nur die ausgewählte Achse wird sich bewegen.
net encoder-counts  <= encoder.0.counts
net encoder-counts => axis.x.jog-counts
net encoder-counts => axis.y.jog-counts
net encoder-counts => axis.z.jog-counts
```

Wenn die Maschine zu einer hohen Beschleunigung fähig ist, um die Bewegungen für jeden Klick des MPG zu glätten, verwenden Sie die Komponente ilowpass, um die Beschleunigung zu begrenzen.

jog.hal mit ilowpass

```
loadrt encoder num_chan=1
loadrt mux4 count=1
addf encoder.capture-position servo-thread
addf encoder.update-counters base-thread
addf mux4.0 servo-thread

loadrt ilowpass
addf ilowpass.0 servo-thread

setp ilowpass.0.scale 1000
setp ilowpass.0.gain 0.01

# Wenn Ihr MPG ein Quadratursignal pro Klick ausgibt, setzen Sie x4 auf 1.
# Wenn Ihr MPG 1 Impuls pro Klick ausgibt, setzen Sie x4 auf 0
setp encoder.0.x4-mode 0

# Für den Geschwindigkeitsmodus, auf 1 setzen.
# Im Geschwindigkeitsmodus hält die Achse an, wenn der Drehknopf gestoppt wird.
# auch wenn das bedeutet, dass die befohlene Bewegung nicht abgeschlossen ist,
# Für den Positionsmodus (die Voreinstellung), setzen Sie auf 0.
# Im Positionsmodus bewegt sich die Achse bei jeder Zählung genau jog-scale viele
# Einheiten für jede Zählung, unabhängig davon, wie lange das dauern könnte,
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0

# Hier wird die Skalierung festgelegt, die auf der Grundlage des Eingangs zum mux4
verwendet wird.
```

```
# Die hier verwendete Skalierung muss mit der ilowpass-Skala multipliziert werden
setp mux4.0.in0 0.0001
setp mux4.0.in1 0.00001
setp mux4.0.in2 0.000001

# Die Eingänge der Komponente mux4
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

# Die Ausgabe des Encoder counts (engl. für Zählerstand) wird an ilowpass gesendet.
net mpg-out ilowpass.0.in <= encoder.0.counts

# Die Ausgabe des mux4 wird an jede Achsen-Jog-Skala gesendet
net mpg-scale <= mux4.0.out
net mpg-scale => axis.x.jog-scale
net mpg-scale => axis.y.jog-scale
net mpg-scale => axis.z.jog-scale

# Die MPG Eingänge
net mpg-a encoder.0.phase-A <= parport.1.pin-02-in
net mpg-b encoder.0.phase-B <= parport.1.pin-03-in

# The Achsen-Auswahl Eingänge
net mpg-x axis.x.jog-enable <= parport.1.pin-04-in
net mpg-y axis.y.jog-enable <= parport.1.pin-05-in
net mpg-z axis.z.jog-enable <= parport.1.pin-06-in

# Die Ausgabe des ilowpasses wird an jede Achsen-Jog-Zählung gesendet
# Nur die ausgewählte Achse wird verschoben.
net encoder-counts  <= ilowpass.0.out
net encoder-counts => axis.x.jog-counts
net encoder-counts => axis.y.jog-counts
net encoder-counts => axis.z.jog-counts
```

7.4. GS2 Spindel

7.4.1. Beispiel

Dieses Beispiel zeigt die Verbindungen benötigt, um eine Automation Direct GS2 VFD verwenden, um eine Spindel zu fahren. Die Spindeldrehzahl und Richtung werden von LinuxCNC gesteuert.

Die Verwendung der GS2-Komponente erfordert nur sehr wenig Einrichtungsaufwand. Wir beginnen mit einer vom StepConf Wizard generierten Konfiguration. Vergewissern Sie sich, dass die Pins mit "Spindle CW" (engl. kurz für Uhrzeigersinn) und "Spindle PWM" im Setup-Bildschirm für den Parallelport auf unbenutzt gesetzt sind.

In der custom.hal Datei platzieren wir das Folgende, um LinuxCNC mit dem GS2 zu verbinden und LinuxCNC den Antrieb steuern zu lassen.

GS2 Beispiel

```
# Laden der nicht-Echtzeit-Komponente für die Automation Direct GS2 VFDs
loadusr -Wn spindle-vfd gs2_vfd -r 9600 -p none -s 2 -n spindle-vfd
```

```
# Verbinden des Pin für die Spindelrichtung mit dem GS2
net gs2-fwd spindle-vfd.spindle-fwd <= spindle.N.forward

# Verbinden des Spindel-Ein-Pin mit dem GS2
net gs2-run spindle-vfd.spindle-on <= spindle.N.on

# Verbinden des GS2 bei Geschwindigkeit mit der Bewegung bei Geschwindigkeit
net gs2-at-speed spindle.N.at-speed <= spindle-vfd.at-speed

# Verbinden der Spindeldrehzahl mit dem GS2
net gs2-RPM spindle-vfd.speed-command <= spindle.N.speed-out
```

NOTE Die Übertragungsgeschwindigkeit kann je nach der genauen Umgebung höher sein. Sowohl das Laufwerk als auch die Befehlszeilenoptionen müssen übereinstimmen. Um auf Übertragungsfehler zu prüfen, fügen Sie die Befehlszeilenoption *-v* hinzu und führen Sie das Programm über ein Terminal aus.

Am GS2-Antrieb selbst müssen Sie einige Einstellungen vornehmen, damit die Modbus-Kommunikation funktioniert. Je nach Ihren physischen Anforderungen müssen möglicherweise noch weitere Parameter eingestellt werden, die jedoch den Rahmen dieses Handbuchs sprengen würden. Weitere Informationen zu den Antriebsparametern finden Sie im GS2-Handbuch, das mit dem Antrieb geliefert wurde.

- Die Kommunikationsschalter müssen auf RS-232C eingestellt sein.
- Die Motorparameter müssen so eingestellt werden, dass sie mit dem Motor übereinstimmen.
- P3.00 (Source of Operation Command) muss auf Operation determined by RS-485 interface, 03 oder 04 gesetzt sein.
- P4.00 (Quelle des Frequenzbefehls) muss auf "Frequenz bestimmt durch RS232C/RS485-Kommunikationsschnittstelle" eingestellt werden, 05.
- P9.01 (Übertragungsgeschwindigkeit) muss auf 9600 Baud eingestellt werden, 01.
- P9.02 (Kommunikationsprotokoll) muss auf "Modbus RTU-Modus, 8 Datenbits, keine Parität, 2 Stoppbits" eingestellt werden, 03.

Ein auf diesem Beispiel basierendes PyVCP-Panel ist hier.

[1] In diesem Beispiel gehen wir davon aus, dass bereits einige Messgeräte an die Achsen/Gelenke 0, 1 und 2 ausgegeben wurden. Das nächste verfügbare Messgerät, das wir an der Spindel anbringen können, wäre also Nummer 3. Ihre Situation kann davon abweichen

[2] Der HAL-Encoder index-enable ist eine Ausnahme von der Regel, da er sich sowohl als Eingang als auch als Ausgang verhält, siehe den Abschnitt zu Encodern für Details

[3] Weil wir oben *non-quadrature simple counting...* ausgewählt haben, können wir mit *quadrature* counting auskommen, ohne einen B-Quadratureingang zu haben.

Chapter 8. ClassicLadder

8.1. ClassicLadder Einführung

8.1.1. Geschichte

ClassicLadder ist eine freie Implementierung eines Ladder/Kontaktplan-Interpreters, veröffentlicht unter der LGPL. Sie wurde von Marc Le Douarain geschrieben.

Er beschreibt den Beginn des Projekts auf seiner Website:

> Ich beschloss, anfangs, im Februar 2001, eine Kontaktplansprache nur zu Testzwecken zu programmieren. Es war geplant, dass ich nach meinem Ausscheiden aus dem Unternehmen, in dem ich damals tätig war, an einem neuen Produkt mitarbeiten würde. Und ich dachte, dass es eine gute Option wäre, eine Ladder Language in diesen Produkten zu haben. Und so begann ich, die ersten Zeilen zu programmieren, um eine Sprosse mit minimalen Elementen zu berechnen und dynamisch unter Gtk darzustellen, um zu sehen, ob meine erste Idee, das alles zu realisieren, funktioniert.
>
> Und da ich schnell festgestellt habe, dass es recht gut vorankommt, habe ich mit komplexeren Elementen weitergemacht: Zeitschaltuhr, mehrere Sprossen usw...
>
> Voila, hier ist diese Arbeit... und mehr: Seitdem habe ich weitere Funktionen hinzugefügt.
>
> — Marc Le Douarain, aus "Genesis" auf der ClassicLadder Website

ClassicLadder wurde angepasst, um mit LinuxCNC's HAL arbeiten, und wird zusammen mit LinuxCNC verteilt. Wenn Fragen/Probleme/Bugs auftreten, melden Sie sie bitte an das LinuxCNC Projekt.

8.1.2. Einführung

Ladder Logic oder die Ladder-Programmiersprache ist eine Methode zum Zeichnen von elektrischen Logikschaltplänen. Sie ist heute eine sehr beliebte grafische Sprache für die Programmierung speicherprogrammierbarer Steuerungen (SPS). Ursprünglich wurde sie erfunden, um eine Logik zu beschreiben, die aus Relais besteht. Der Name beruht auf der Beobachtung, dass Programme in dieser Sprache Leitern ähneln, mit zwei vertikalen "Schienen" und einer Reihe von horizontalen "Sprossen" dazwischen. In Deutschland und anderswo in Europa ist es üblich, die Sprossen waagerecht am oberen und unteren Rand der Seite zu zeichnen, während die Sprossen senkrecht von links nach rechts verlaufen.

Ein Programm in Kontaktplanlogik, auch Kontaktplan genannt, ist einem Schaltplan für eine Reihe von Relaisschaltungen ähnlich. Die Kontaktplanlogik ist nützlich, weil eine Vielzahl von Ingenieuren und Technikern sie aufgrund dieser Ähnlichkeit ohne viel zusätzliche Schulung verstehen und verwenden kann.

Die Kontaktplanlogik wird häufig zur Programmierung von SPS verwendet, wenn eine sequenzielle Steuerung eines Prozesses oder eines Fertigungsvorgangs erforderlich ist. Die Kontaktplanlogik ist nützlich für einfache, aber kritische Steuerungssysteme oder für die Überarbeitung alter festverdrahteter Relaisschaltungen. Da speicherprogrammierbare Steuerungen immer ausgereifter wurden, wurde sie auch in sehr komplexen Automatisierungssystemen eingesetzt.

Die Kontaktplanlogik kann als regelbasierte Sprache und nicht als prozedurale Sprache betrachtet werden. Eine "Sprosse" im Kontaktplan stellt eine Regel dar. Bei der Verwendung von Relais und anderen elektromechanischen Geräten werden die verschiedenen Regeln gleichzeitig und sofort "ausgeführt". Bei der Implementierung in eine speicherprogrammierbare Steuerung werden die Regeln in der Regel sequentiell von der Software in einer Schleife ausgeführt. Wenn die Schleife schnell genug ausgeführt wird, in der Regel viele Male pro Sekunde, wird der Effekt der gleichzeitigen und unmittelbaren Ausführung erzielt.

Die Kontaktplanlogik folgt diesen allgemeinen Schritten für den Betrieb.

- Eingänge lesen
- Logik auflösen
- Ausgaben aktualisieren

8.1.3. Beispiel

Die gebräuchlichsten Bestandteile von Leitern sind Kontakte (Eingänge), die in der Regel entweder Öffner (NC) oder Schließer (NO) sind, und Spulen (Ausgänge).

- der Schließer 
- der Öffner 
- die Spule (Ausgang) 

Natürlich gibt es noch viele weitere Komponenten einer vollständigen Leitersprache, aber das Verständnis dieser Komponenten wird Ihnen helfen, das Gesamtkonzept zu begreifen.

Die Leiter eines Kontaktplans besteht aus einer oder mehreren Sprossen. Diese Sprossen sind horizontale Leiterbahnen (die Drähte darstellen), auf denen sich Komponenten befinden (Eingänge, Ausgänge und andere), die von links nach rechts ausgewertet werden.

Dieses Beispiel ist die einfachste Sprosse:



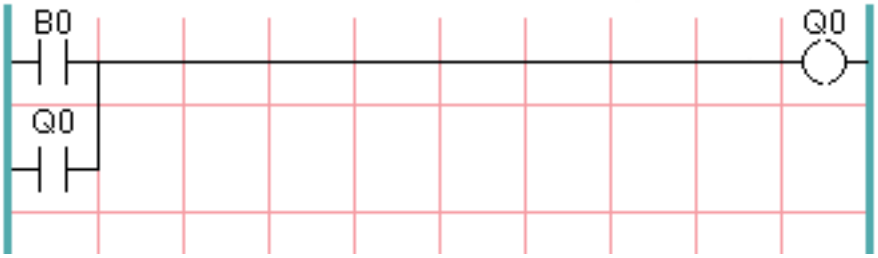
Der Eingang auf der linken Seite, B_0, ein Schließer, ist mit der Spule (Ausgang) auf der rechten Seite, Q_0, verbunden. Stellen Sie sich nun vor, dass am linken Ende eine Spannung angelegt wird, weil der Eingang B_0 aktiv wird (z. B. weil der Eingang aktiviert ist oder der Benutzer den Schließer betätigt hat). Die Spannung hat einen direkten Weg zur Spule (Ausgang) rechts, Q_0. Infolgedessen schaltet die Spule Q_0 (Ausgang) von 0/aus/falsch auf 1/an/wahr. Wenn der Benutzer B_0 loslässt, kehrt der Ausgang Q_0 schnell zu 0/aus/falsch zurück.

8.1.4. Grundlegende selbsthaltende Ein-Aus-Schaltung

Nehmen wir an, wir fügen dem obigen Beispiel einen Schalter hinzu, der immer dann schließt, wenn die Spule Q_0 aktiv ist. Dies wäre der Fall bei einem Relais, bei dem die Spule die Schaltkontakte aktivieren kann, oder bei einem Schütz, bei dem es oft mehrere kleine Hilfskontakte zusätzlich zu den großen dreiphasigen Kontakten gibt, die das Hauptmerkmal des Schützes sind.

Da dieser Hilfsschalter in unserem früheren Beispiel von der Spule Q_0 angesteuert wird, geben wir ihm die gleiche Nummer wie der Spule, die ihn ansteuert. Dies ist die Standardpraxis in der Kontaktplanprogrammierung, auch wenn es zunächst seltsam erscheinen mag, dass ein Schalter mit der gleichen Nummer wie eine Spule bezeichnet wird. Nennen wir also diesen Hilfskontakt Q_0 und verbinden ihn mit dem Tasterkontakt B_0 aus unserem früheren Beispiel.

Werfen wir einen Blick darauf:



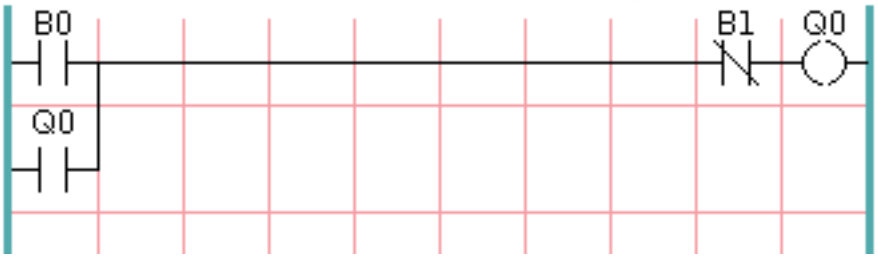
Wenn der Benutzer den Taster B0 drückt, schaltet sich wie zuvor die Spule Q0 ein. Und wenn die Spule Q0 eingeschaltet wird, dann wird der Schalter Q0 eingeschaltet. Jetzt kommt der interessante Teil. Wenn der Benutzer den Druckknopf B0 loslässt, bleibt die Spule Q0 nicht wie zuvor stehen. Das liegt daran, dass der Schalter Q0 dieser Schaltung den Druckknopf des Benutzers tatsächlich gedrückt hält. Wir sehen also, dass der Schalter Q0 die Spule Q0 auch dann noch anhält, wenn der Starttaster losgelassen wurde.

Diese Art von Kontakt an einer Spule oder einem Relais, auf diese Weise verwendet, wird oft als "Haltekontakt" bezeichnet, weil er die Spule, mit der er verbunden ist, "festhält". Gelegentlich wird er auch als "Siegelkontakt" (engl. seal contact) bezeichnet, und wenn er aktiv ist, sagt man, dass der Stromkreis "versiegelt" ist.

Leider hat unsere Schaltung bisher wenig praktischen Nutzen, denn wir haben zwar einen Einschalt- oder Startknopf in Form des Tasters B_0, aber wir haben keine Möglichkeit, diese Schaltung abzuschalten, wenn sie einmal gestartet ist. Aber das ist leicht zu beheben. Alles, was wir brauchen, ist eine

Möglichkeit, die Stromzufuhr zur Spule Q_0 zu unterbrechen. Fügen wir also einen Öffner-Taster direkt vor der Spule Q_0 ein.

Das würde folgendermaßen aussehen:



Jetzt haben wir den Taster B_1 für "Aus" oder "Stopp" hinzugefügt. Wenn der Benutzer ihn drückt, wird der Kontakt zwischen der Sprosse und der Spule unterbrochen. Wenn die Spule Q_0 keinen Strom mehr hat, geht sie auf 0/aus/falsch. Wenn die Spule Q_0 erlischt, dann erlischt auch der Schalter Q_0, so dass der "Haltekontakt" unterbrochen oder der Stromkreis "entsiegelt" wird. Wenn der Benutzer die Stopptaste loslässt, wird der Kontakt von der Sprosse zur Spule Q_0 wiederhergestellt, aber die Sprosse ist erloschen, so dass die Spule nicht wieder eingeschaltet wird.

Diese Schaltung wird seit Jahrzehnten in praktisch jeder Maschine verwendet, die einen durch ein Schütz gesteuerten Drehstrommotor hat, so dass es unvermeidlich war, dass sie von Kontaktplan-/SPS-Programmierern übernommen werden würde. Es handelt sich auch um eine sehr sichere Schaltung, denn wenn "Start" und "Stopp" gleichzeitig gedrückt werden, gewinnt immer die "Stopp"-Funktion.

Dies ist der Grundbaustein eines Großteils der Kontaktplanprogrammierung. Wenn Sie also neu in diesem Bereich sind, sollten Sie sicherstellen, dass Sie verstehen, wie diese Schaltung funktioniert.

8.2. ClassicLadder Programmierung

8.2.1. SPS / Kontaktplan Konzepte

ClassicLadder ist eine Art Programmiersprache, die ursprünglich auf industriellen SPS implementiert wurde (sie wird Kontaktplan Programmierung genannt, engl. der Ähnlichkeit zu Leitern halber *ladder programming*) . Sie basiert auf dem Konzept der Relaiskontakte und -spulen und kann verwendet werden, um logische Prüfungen und Funktionen auf eine Weise zu konstruieren, die vielen Systemintegratoren vertraut ist. Der Kontaktplan besteht aus Sprossen, die sich verzweigen können, und ähnelt einem elektrischen Schaltkreis. Es ist wichtig zu wissen, wie Kontaktplanprogramme bei der Ausführung ausgewertet werden.

Entsprechend unserer Gewohnheiten würden wir erwarten, dass jede Zeile von links nach rechts ausgewertet wird, dann die nächste Zeile nach unten usw., aber in der Kontaktplanlogik funktioniert das nicht so. In der Kontaktplanlogik werden die Kontaktsprossen 3 Mal "gescannt", um den Zustand der Ausgänge zu ändern.

- die Eingänge werden gelesen und aktualisiert
- die Logik wird abgeleitet

- die Ausgänge (engl. outputs) werden gesetzt

Dies kann zunächst verwirrend sein, wenn der Ausgang einer Zeile durch den Eingang eines anderen Rings gelesen wird. Es wird eine Abfrage geben, bevor der zweite Eingang wahr wird, nachdem der Ausgang gesetzt wurde.

Ein weiteres Problem bei der Kontaktplanprogrammierung ist die "Last One Wins"-Regel. Wenn Sie denselben Ausgang an verschiedenen Stellen Ihres Kontaktplans haben, wird der Zustand des letzten Ausgangs derjenige sein, auf den der Ausgang eingestellt ist.

8.2.2. Sprachen

Die gebräuchlichste Sprache bei der Arbeit mit ClassicLadder ist der "Kontaktplan" (engl. ladder). ClassicLadder unterstützt auch Ablaufpläne (Grafcet).

8.2.3. Komponenten

ClassicLadder besteht aus zwei Komponenten.

- Das Echtzeit-Modul classicladder_rt
- Das Nicht-Echtzeit-Modul (einschließlich einer grafischen Benutzeroberfläche) classicladder

Dateien

Normalerweise werden klassische Kontaktplan-Komponenten in der Datei custom.hal platziert, wenn Sie mit einer von StepConf generierten Konfiguration arbeiten. Diese dürfen nicht in der Datei custom_postgui.hal platziert werden, da sonst das Kontaktplan-Editor-Menü ausgegraut wird.

NOTE Kontaktplan-Dateien (.clp) dürfen keine Leerzeichen im Namen enthalten.

Echtzeit-Modul

Das Laden des ClassicLadder-Echtzeitmoduls (classicladder_rt) ist aus einer HAL-Datei oder direkt mit einem halcmd-Befehl möglich. Die erste Zeile lädt das ClassicLadder-Modul in Echtzeit. Die zweite Zeile fügt die Funktion classicladder.0.refresh in den Servo-Thread ein. Diese Zeile bewirkt, dass ClassicLadder mit der Rate des Servo-Threads aktualisiert wird.

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

Die Geschwindigkeit des Threads, in dem ClassicLadder läuft, wirkt sich direkt auf die Reaktionsfähigkeit auf Eingaben und Ausgaben aus. Wenn Sie einen Schalter schneller ein- und ausschalten können, als ClassicLadder ihn wahrnehmen kann, müssen Sie den Thread möglicherweise beschleunigen. Die schnellste Zeit, in der ClassicLadder die Sprossen aktualisieren kann, ist eine Millisekunde. Sie können ihn in einen schnelleren Thread setzen, aber er wird nicht schneller aktualisieren. Wenn Sie einen langsameren Thread als eine Millisekunde verwenden, wird ClassicLadder die Sprossen langsamer aktualisieren. Die aktuelle Abfragezeit wird in der

Abschnittsanzeige angezeigt, sie wird auf Mikrosekunden gerundet. Wenn die Abfragezeit länger als eine Millisekunde ist, sollten Sie den Kontaktplan verkürzen oder in einen langsameren Thread einfügen.

Variablen

Es ist möglich, die Anzahl der einzelnen Arten von Kontaktplanobjekten beim Laden des ClassicLadder Echtzeitmoduls zu konfigurieren. Wenn Sie die Anzahl der Kontaktplanobjekte nicht konfigurieren, verwendet ClassicLadder die Standardwerte.

Table 41. Standard-Variablenzahl

Objektname	Variablenname	Standardwert
Anzahl der Sprossen	(numRungs)	100
Anzahl der Bits	(numBits)	20
Anzahl der Wortvariablen	(numWords)	20
Anzahl der Timer	(numTimers)	10
Anzahl der Timer IEC	(numTimersIec)	10
Anzahl der Monostabilen	(numMonostables)	10
Anzahl der Zähler	(numCounters)	10
Number of HAL inputs bool pins	(numPhysInputs)	15
Number of HAL output bool pins	(numPhysOutputs)	15
Anzahl der arithmetischen Ausdrücke	(numArithmExpr)	50
Anzahl der Abschnitte	(numSections)	10
Anzahl der Symbole	(numSymbols)	Auto
Number of Sint inputs	(numS32in)	10
Number of Sint outputs	(numS32out)	10
Number of Real inputs	(numFloatIn)	10
Number of Real outputs	(numFloatOut)	10

Objekte von größtem Interesse sind numPhysInputs, numPhysOutputs, numS32in und numS32out.

Changing these numbers will change the number of HAL bool pins available. numPhysInputs and numPhysOutputs control how many HAL bool (on/off) pins are available. numS32in and numS32out control how many HAL signed integers (+- integer range) pins are available.

Zum Beispiel (Sie brauchen nicht alle, nur einige wenige zu ändern):

```
loadrt classicladder_rt numRungs=12 numBits=100 numWords=10
```

```
numTimers=10 numMonostables=10 numCounters=10 numPhysInputs=10
numPhysOutputs=10 numArithmExpr=100 numSections=4 numSymbols=200
numS32in=5 numS32out=5
```

Um die Standardanzahl von Objekten zu laden:

```
loadrt classicladder_rt
```

8.2.4. Laden des ClassicLadder Nicht-Echtzeit-Moduls

ClassicLadder-HAL-Befehle müssen ausgeführt werden, bevor die GUI geladen wird, sonst funktioniert der Menüpunkt Kontaktplan-Editor nicht. Wenn Sie den Stepper Config Wizard verwendet haben, platzieren Sie alle ClassicLadder-HAL-Befehle in der Datei custom.hal.

Diese Zeile lädt das Nicht-Echtzeit-Modul:

```
loadusr classicladder
```

NOTE Es kann nur eine .clp-Datei geladen werden. Wenn Sie Ihren Kontaktplan unterteilen müssen, dann verwenden Sie Abschnitte.

Um eine Kontaktplan-Datei zu laden:

```
loadusr classicladder myladder.clp
```

ClassicLadder Lade-Optionen

- --nogui' - (lädt ohne den Kontaktplan-Editor) wird normalerweise verwendet, wenn die Fehlersuche beendet ist.
- *--modbus_port=port* - (lädt die Modbus-Portnummer)
- *--modmaster* - (initialisiert den MODBUS-Master) sollte das Kontaktplanprogramm zur gleichen Zeit laden oder der TCP-Port ist Standard.
- *--modslave* - (initialisiert MODBUS-Slave) nur TCP

Zur Verwendung von ClassicLadder mit HAL ohne EMC:

```
loadusr -w classicladder
```

Die Option -w weist HAL an, die HAL-Umgebung nicht zu schließen, bevor Classic Ladder beendet ist.

Wenn Sie zuerst ein Kontaktplanprogramm mit der Option *--nogui* laden und dann ClassicLadder erneut ohne Optionen laden, zeigt die GUI das zuletzt geladene Kontaktplanprogramm an.

In AXIS können Sie die GUI über Datei/Kontaktplan-Editor... laden.

8.2.5. ClassicLadder GUI

Wenn Sie ClassicLadder mit der GUI laden, werden zwei Fenster angezeigt: Abschnittsanzeige und Abschnittsmanager.

Sektions-Manager

Wenn Sie ClassicLadder zum ersten Mal starten, sehen Sie ein leeres Fenster des Abschnittsmanagers.

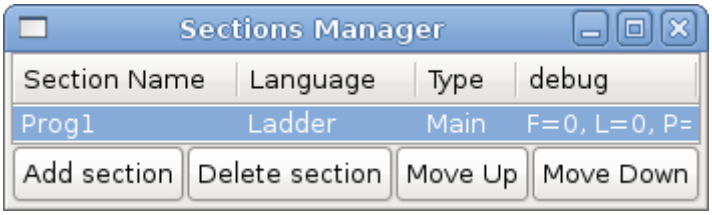


Figure 110. Sections Manager Standardfenster

In diesem Fenster können Sie Abschnitte benennen, erstellen oder löschen und auswählen, welche Sprache der Abschnitt verwendet. Auf diese Weise benennen Sie auch ein Unterprogramm für Aufrufspulen (engl. call coils).

Abschnittsanzeige

Wenn Sie ClassicLadder zum ersten Mal starten, sehen Sie ein leeres Abschnittsanzeigefenster. Es wird eine leere Sprosse angezeigt.

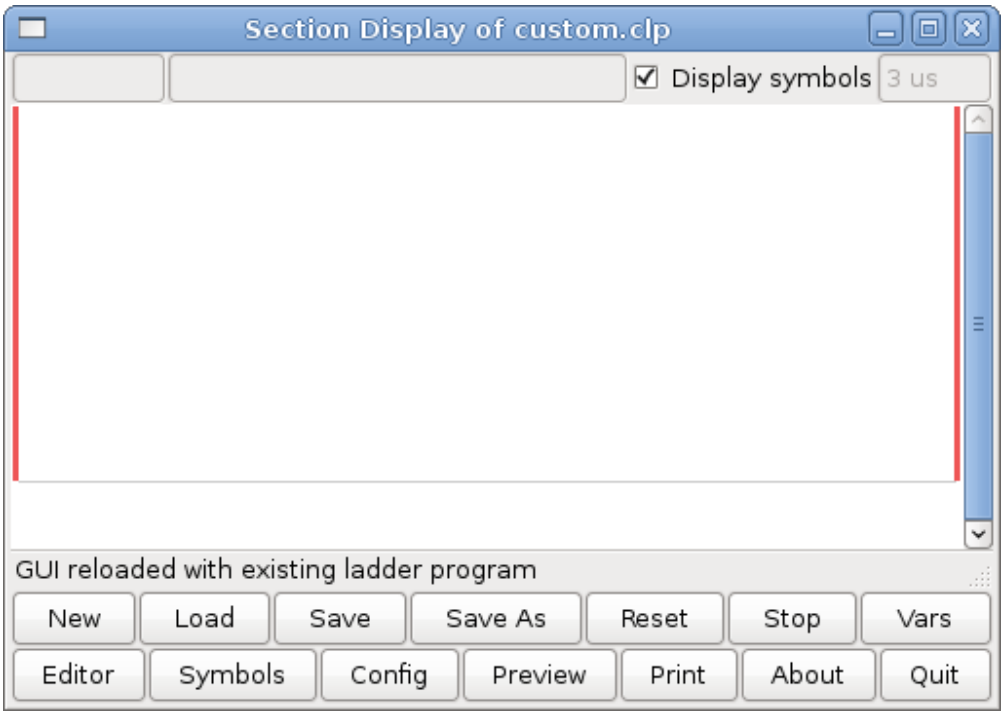


Figure 111. Standardfenster der Abschnittsanzeige

Die meisten Buttons sind selbsterklärend:

Die Button Vars dient zur Anzeige von Variablen. Sie können sie umschalten, um das eine, das andere, beide oder keines der Fenster anzuzeigen.

Der Button Config wird für Modbus verwendet und zeigt die maximale Anzahl von Kontaktplanelementen an, die mit dem Echtzeitmodul geladen wurden.

Der Button Symbole zeigt eine editierbare Liste von Symbolen für die Variablen an (Hinweis: Sie können die Eingänge, Ausgänge, Spulen usw. benennen).

Mit der Schaltfläche Beenden wird das Nicht-Echtzeitprogramm, d. h. Modbus und das Display, beendet. Das Echtzeit-Kontaktplanprogramm wird weiterhin im Hintergrund ausgeführt.

Über das Kontrollkästchen oben rechts können Sie auswählen, ob Variablennamen oder Symbolnamen angezeigt werden

Vielleicht fällt Ihnen auf, dass unter der Anzeige des Kontaktplanprogramms eine Zeile mit der Aufschrift "Projekt konnte nicht geladen werden..." zu sehen ist. Das ist die Statusleiste, die Ihnen Informationen über Elemente des Kontaktplanprogramms gibt, auf die Sie im Anzeigefenster klicken. In dieser Statuszeile werden nun die HAL-Signalnamen für die Variablen %I, %Q und das erste %W (in einer Gleichung) angezeigt. Möglicherweise sehen Sie einige lustige Bezeichnungen, wie (103) in den Sprossen. Dies wird (absichtlich) aufgrund eines alten Fehlers angezeigt - beim Löschen von Elementen löschten ältere Versionen manchmal das Objekt nicht mit dem richtigen Code. Vielleicht haben Sie bemerkt, dass die lange horizontale Verbindungstaste in älteren Versionen manchmal nicht funktionierte. Das lag daran, dass das Programm nach dem *freien* Code suchte, aber etwas anderes fand. Die Zahl in den Klammern ist der nicht erkannte Code. Das Kontaktplanprogramm funktioniert trotzdem, löschen Sie die Codes mit dem Editor und speichern Sie das Programm.

Die Variablenfenster

Dies sind zwei Variablenfenster: das Bitstatus-Fenster (boolesch) und das Watch-Fenster (vorzeichenbehaftete Ganzzahl). Die Schaltfläche "Variablen" befindet sich im Fenster "Abschnittsanzeige". Wechseln Sie mit dem Button "Variablen" die Anzeige, um das eine, das andere, beide oder keines der Variablenfenster anzuzeigen.

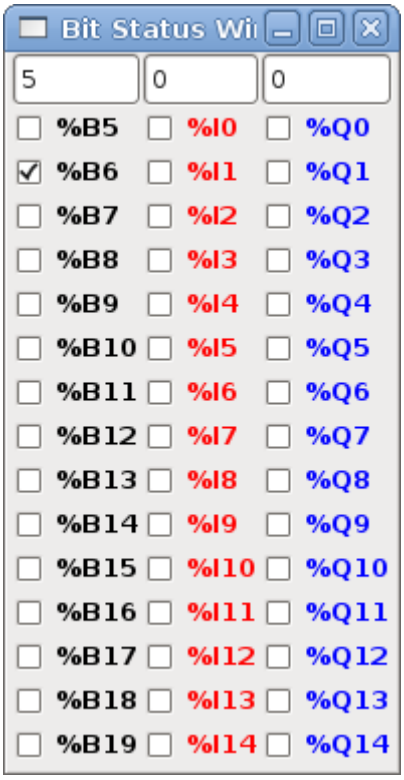


Figure 112. Bit-Status-Fenster

The Bit Status Window displays some of the boolean (on/off) variable data. Notice all variables start with the % sign. The %I variables represent HAL input bool pins. The %Q represents the relay coil and HAL output bool pins. The %B represents an internal relay coil or internal contact. The three edit areas at the top allow you to select what 15 variables will be displayed in each column. For instance, if the %B Variable column were 15 entries high, and you entered 5 at the top of the column, variables %B5 to %B19 would be displayed. The check boxes allow you to set and unset %B variables manually as long as the ladder program isn't setting them as outputs. Any Bits that are set as outputs by the program when ClassicLadder is running can not be changed and will be displayed as checked if on and unchecked if off.

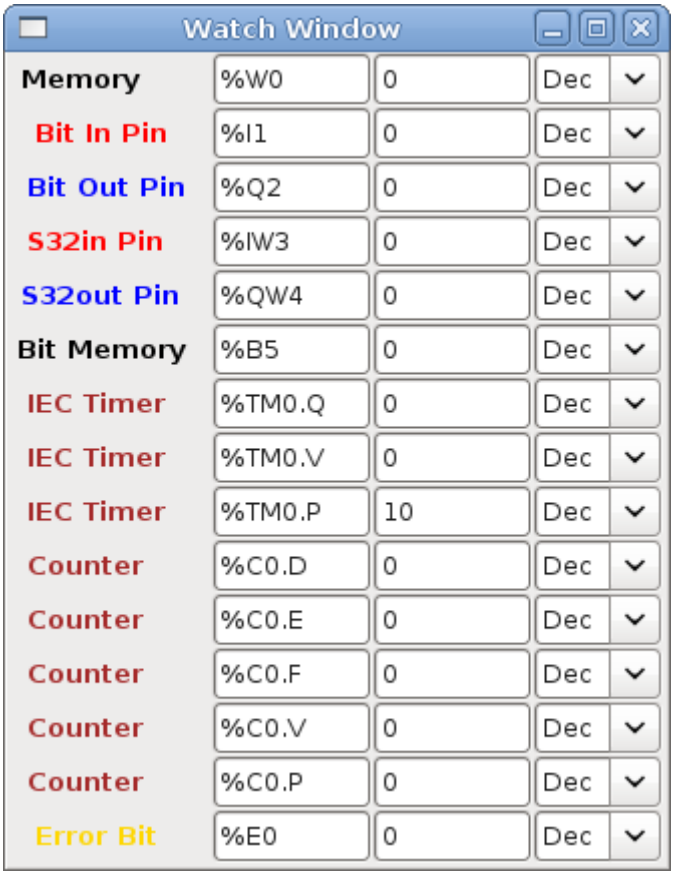


Figure 113. Schaufenster

Das Watch Window zeigt den Status der Variablen an. Das Eingabefeld daneben zeigt die in der Variablen gespeicherte Zahl an, und in der Dropdown-Box daneben können Sie wählen, ob die Zahl in Hex, Dezimal oder Binär angezeigt werden soll. Wenn im Symbole-Fenster Symbolnamen für die angezeigten Wortvariablen definiert sind und das Kontrollkästchen "Symbole anzeigen" im Fenster "Abschnitt anzeigen" aktiviert ist, werden die Symbolnamen angezeigt. Um die angezeigte Variable zu ändern, geben Sie die Variablennummer ein, z. B. %W2 (wenn das Kontrollkästchen "Symbole anzeigen" nicht aktiviert ist), oder geben Sie den Symbolnamen (wenn das Kontrollkästchen "Symbole anzeigen" aktiviert ist) über eine bestehende Variablennummer/einen bestehenden Variablennamen ein und drücken Sie die Eingabetaste.

Symbol-Fenster

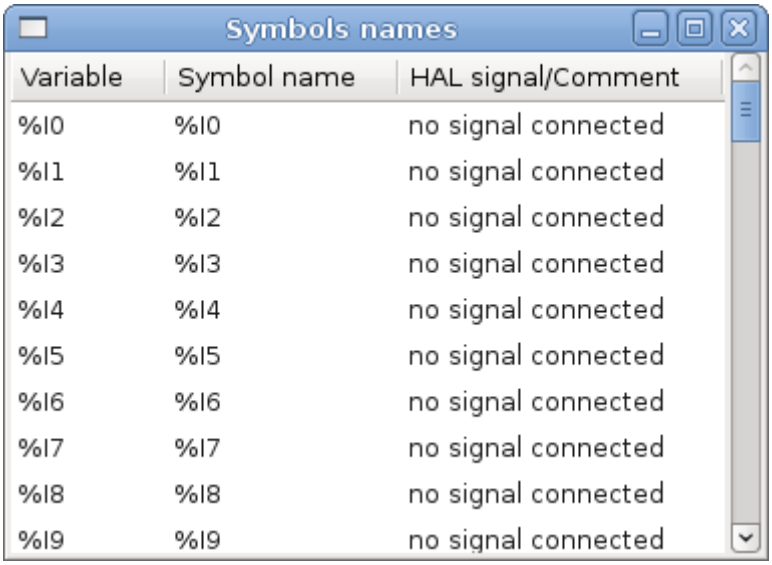


Figure 114. Fenster mit Symbolnamen

This is a list of *symbol* names to use instead of variable names to be displayed in the section window when the *display symbols* check box is checked. You add the variable name (remember the *%* symbol and capital letters), symbol name. If the variable can have a HAL signal connected to it (%I, %Q, and %W-if you have loaded sint pin with the real time module) then the comment section will show the current HAL signal name or lack thereof. Symbol names should be kept short to display better. Keep in mind that you can display the longer HAL signal names of %I, %Q and %W variable by clicking on them in the section window. Between the two, one should be able to keep track of what the ladder program is connected to!

Das Editor-Fenster

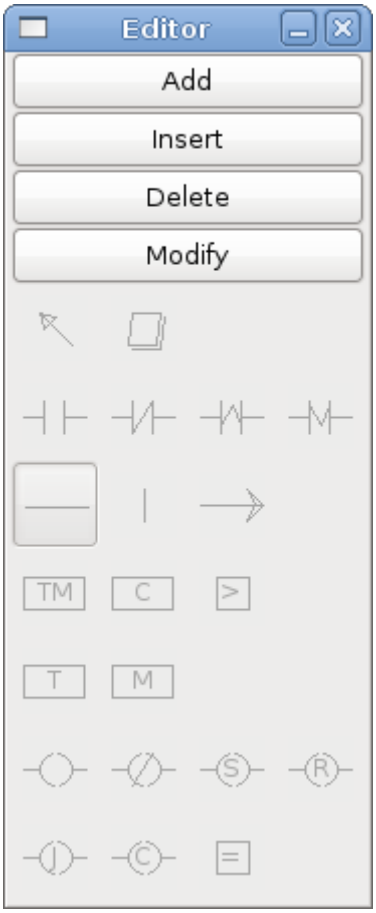


Figure 115. Editor-Fenster

- *Hinzufügen* (engl. add) - fügt eine Sprosse nach der ausgewählten Sprosse hinzu
- *Einfügen* (engl. insert) - fügt eine Sprosse vor der ausgewählten Sprosse ein
- *Löschen* (engl. delete) - löscht die ausgewählte Sprosse
- *Bearbeiten* (engl. modify) - öffnet die ausgewählte Sprosse zur Bearbeitung

Beginnend mit dem Bild oben links:

- Objektauswahl, Radiergummi
- N.O. (engl. kurz für *normally open*)-Eingang, N.C. (engl. für normally closed)-Eingang, Eingang mit steigender Flanke, Eingang mit fallender Flanke
- Horizontale Verbindung, Vertikale Verbindung, Lange Horizontale Verbindung
- Timer IEC-Block, Zähler (engl. counter-)block, Vergleichsvariable
- Alter Timer-Block, alter monostabiler Block (diese wurden durch den IEC-Timer ersetzt)
- SPULEN (engl. coils) - N.O. Ausgang, N.C. Ausgang, den Ausgang setzen, Ausgang zurücksetzen
- Jump Coil, Call Coil, Variable Zuweisung

Eine kurze Beschreibung der einzelnen Buttons:

- *Selector* - ermöglicht es Ihnen, vorhandene Objekte auszuwählen und die Informationen zu ändern.
- *Radiergummi* (engl. eraser) - löscht ein Objekt.

- *N.O. Contact* - erzeugt einen normalerweise offenen Kontakt. Es kann sich um einen externen HAL-Pin (%I) Eingangskontakt, einen internen Bitspulenkontakt (%B) oder einen externen Spulenkontakt (%Q) handeln. Der HAL-Pin-Eingangskontakt wird geschlossen, wenn der HAL-Pin true ist. Die Spulenkontakte werden geschlossen, wenn die entsprechende Spule aktiv ist (%Q2-Kontakt schließt sich, wenn %Q2-Spule aktiv ist).
- *N.C. Kontakt* - erstellt einen normalerweise geschlossenen Kontakt. Es ist das gleiche wie der N.O. Kontakt, außer dass der Kontakt offen ist, wenn der HAL-Pin wahr ist oder die Spule aktiv ist.
- *Rising Edge Contact* - erzeugt einen Kontakt, der geschlossen wird, wenn der HAL-Pin von False auf True oder die Spule von nicht-aktiv auf aktiv wechselt.
- *Falling Edge Contact* - erzeugt einen Kontakt, der geschlossen wird, wenn der HAL-Pin von true nach false oder die Spule von aktiv zu not wechselt.
- *Horizontale Verbindung* - erzeugt eine horizontale Verbindung zu Objekten.
- *Vertikale Verbindung* - erzeugt eine vertikale Verbindung zu horizontalen Linien.
- *Horizontal Running Connection* - erstellt eine horizontale Verbindung zwischen zwei Objekten und ist eine schnelle Möglichkeit, Objekte zu verbinden, die mehr als einen Block voneinander entfernt sind.
- *IEC Timer* - erstellt einen Timer und ersetzt den *Timer*.
- *Timer* - erstellt ein Timer-Modul (veraltet, verwenden Sie stattdessen IEC-Timer).
- *Monostabil* - erstellt ein monostabiles One-Shot-Modul
- *Counter* - erstellt ein Zählermodul.
- *Compare* - erstellt einen Vergleichsblock, um Variablen mit Werten oder anderen Variablen zu vergleichen, z.B. `%W1<=5` oder `%W1=%W2`. Der Vergleich kann nicht auf der rechten Seite der Abschnittsanzeige platziert werden.
- *Variablenzuweisung* (engl. variable assignment) - erstellt einen Zuweisungsblock, damit Sie Variablen Werte zuweisen können, z.B. `%W2=7` oder `%W1=%W2`. ASSIGNMENT-Funktionen können nur auf der rechten Seite der Abschnittsanzeige platziert werden.

Konfigurationsfenster

Das Konfigurationsfenster zeigt den aktuellen Projektstatus und enthält die Registerkarten für die Modbus-Einrichtung.

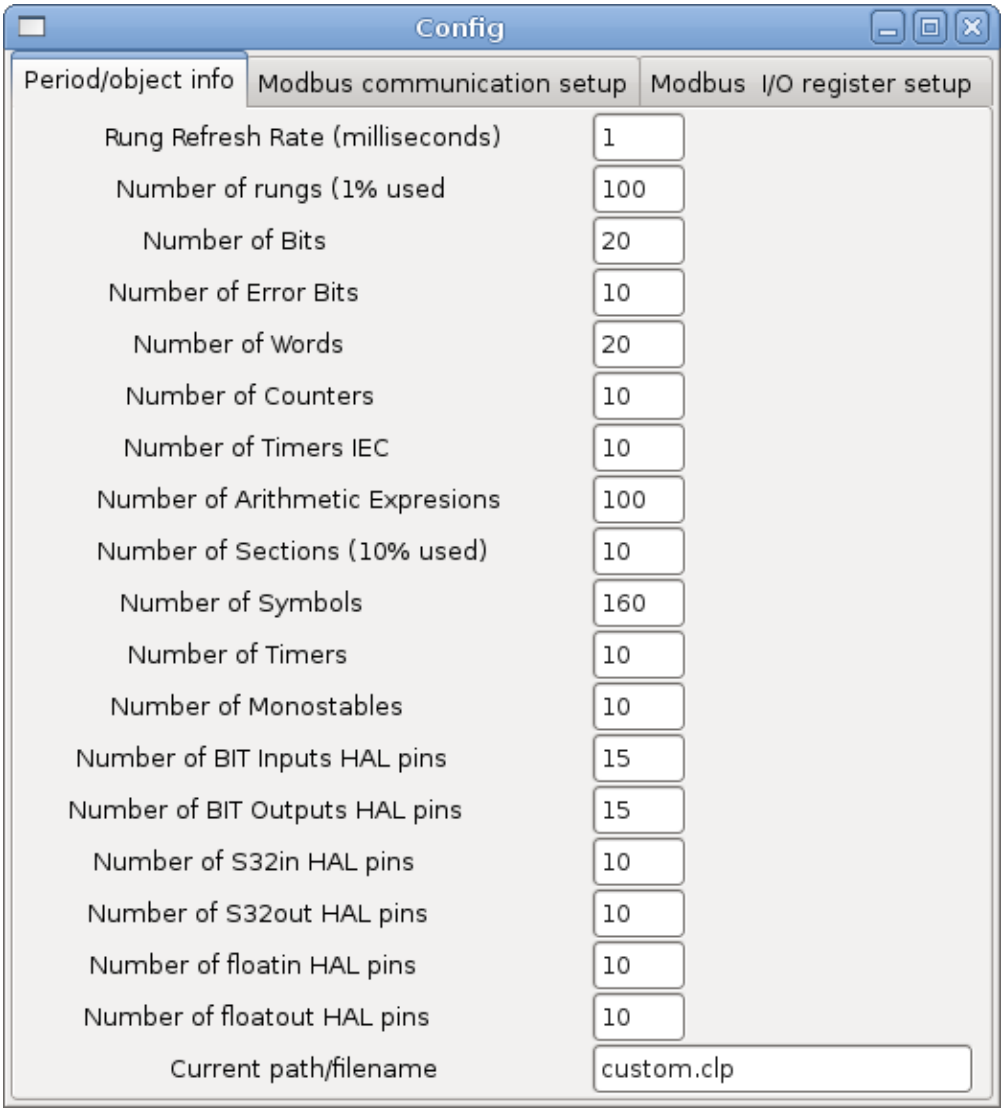


Figure 116. Konfigurationsfenster

8.2.6. SPS Objekte

KONTAKTE

Stellen Schalter oder Relaiskontakte dar. Sie werden durch den ihnen zugewiesenen variablen Buchstaben und die Nummer gesteuert.

Der variable Buchstabe kann B, I oder Q sein und die Nummer kann bis zu einer dreistelligen Zahl sein, z. B. `%I2`, `%Q3` oder `%B123`. Die Variable I wird durch einen HAL-Eingangsstift mit einer entsprechenden Nummer gesteuert. Die Variable B ist für interne Kontakte und wird von einer B-Spule mit einer entsprechenden Nummer gesteuert. Die Variable Q wird durch eine Q-Spule mit der entsprechenden Nummer gesteuert (wie ein Relais mit mehreren Kontakten). Wenn z. B. der HAL-Pin `classicladder.0.in-00` auf true steht, ist der Schließerkontakt %I0 eingeschaltet (geschlossen, true, wie auch immer Sie es nennen wollen). Wenn die Spule %B7 "erregt" ist (ein, wahr, usw.), dann wäre der Schließerkontakt %B7 eingeschaltet. Wenn die Spule %Q1 "erregt" ist, wäre der Schließer %Q1 eingeschaltet (und der HAL-Pin `classicladder.0.out-01` wäre wahr).

- N.O. Contact' - ⊣ ⊢ (Schließer) Ist die Variable mit false belegt, dann ist der Schalter aus.

- *N.C. Contact* -  (Normalerweise geschlossen) Ist die Variable auf false gesetzt, so ist der Schalter eingeschaltet.
- *Rising Edge Contact* - Wenn die Variable von false in true wechselt, wird der Schalter gepulst.
- *Falling Edge Contact* - Wenn die Variable von true zu false wechselt, wird der Schalter gepulst eingeschaltet.

IEC-TIMER

Stellt neue Countdown-Timer dar. IEC-Timer ersetzen Timer und Monoflops.

IEC Timer haben 2 Kontakte.

- *I* - Eingabekontakt
- *Q* - Ausgangskontakt

Es gibt drei Modi - TON, TOF, TP.

- *TON* - Wenn die Timer-Eingabe wahr ist, beginnt der Countdown und wird fortgesetzt, solange die Eingabe wahr bleibt. Nachdem der Countdown abgeschlossen ist und solange die Timer-Eingabe noch wahr, ist die Ausgabe wahr.
- *TOF* - Wenn die Timereingabe true ist, wird die Ausgabe auf true gesetzt. Wenn die Eingabe false ist, zählt der Timer herunter und setzt die Ausgabe auf false.
- *TP* - Wenn der Timer-Eingang auf wahr gepulst oder wahr gehalten wird, setzt der Timer den Ausgang auf wahr, bis der Timer herunterzählt. (einmalig)

Die Zeitintervalle können in Vielfachen von 100 ms, Sekunden oder Minuten eingestellt werden.

Es gibt auch Variablen für IEC-Timer, die in Vergleichs- oder Betriebsblöcken gelesen und/oder beschrieben werden können.

- *%TMxxx.Q* - Timer beendet (Boolesch, Lesen/Schreiben)
- *%TMxxx.P* - Timer-Voreinstellung (Lesen/Schreiben)
- *%TMxxx.V* - Timer-Wert (lesender Schreibvorgang)

ZEITGLIEDER (engl. timers)

Repräsentiert Countdown-Timer. Dies ist veraltet und wird durch IEC Timers ersetzt.

Timer haben 4 Kontakte.

- *E* - aktivieren (engl. enable) (Eingang) startet den Timer, wenn wahr, setzt zurück, wenn er falsch wird
- *C* - Steuerung (engl. control) (Eingang) muss eingeschaltet sein, damit der Timer läuft (normalerweise mit E verbinden)
- *D* - fertig (engl. done) (Ausgang) wahr, wenn der Timer abläuft und solange E wahr bleibt

- *R* - läuft (engl. running) (Ausgang) true, wenn Timer läuft

Die Basis des Timers kann ein Vielfaches von Millisekunden, Sekunden oder Minuten sein.

Es gibt auch Variablen für Zeitgeber, die in Vergleichs- oder Operationsblöcken gelesen und/oder beschrieben werden können.

- *%Txx.R* - Timer xx läuft (boolesch, nur Lesen)
- *%Txx.D* - Timer xx fertig (Boolesch, nur lesbar)
- *%Txx.V* - Timer xx aktueller Wert (Ganzzahl, nur lesbar)
- *%Txx.P* - Timer xx voreingestellt (Ganzzahl, lesen oder schreiben)

KIPPSTUFEN (engl. monostables)

Repräsentieren die ursprünglichen One-Shot-Timer. Dies ist jetzt veraltet und wird durch IEC-Timer ersetzt.

Monostabile haben 2 Kontakte, I und R.

- *I* - Eingang (Eingang) startet den Mono-Timer.
- *R* - Running (Ausgang) ist wahr, während der Timer läuft.

Der I-Kontakt reagiert auf eine steigende Flanke, d.h. er startet den Timer nur, wenn er von false auf true (oder von off auf on) wechselt. Während der Timer läuft, kann sich der I-Kontakt ändern, ohne dass dies Auswirkungen auf den laufenden Timer hat. R wird wahr und bleibt wahr, bis der Timer auf Null gezählt hat. Die Basis des Timers kann ein Vielfaches von Millisekunden, Sekunden oder Minuten sein.

Es gibt auch Variablen für Kippstufen, die in Vergleichs- oder Operationsblöcken gelesen und/oder beschrieben werden können.

- *%Mxx.R* - Kippstufe xx läuft (boolesch, nur lesbar)
- *%Mxx.V* - Monostabiler xx aktueller Wert (ganze Zahl, schreibgeschützt)
- *%Mxx.P* - Monostabile xx-Voreinstellung (Ganzzahl, Lesen oder Schreiben)

ZÄHLER (engl. counters)

Aufwärts-/Abwärtszähler darstellen.

Es gibt 7 Kontakte:

- *R* - reset (Eingabe) setzt die Anzahl auf 0 zurück.
- *P* - Voreinstellung (engl. preset) (Eingabe) setzt den Zähler auf die im Bearbeitungsmenü zugewiesene Voreinstellungsnummer.
- *U* - Aufwärtszählung (Eingabe) fügt der Zählung eins hinzu.
- *D* - Abwärtszählung (Eingang) subtrahiert eins von der Zählung.

- *E* - under flow (output) ist wahr, wenn die Zählung von 0 auf 9999 übertragen wird.
- *D* - done (Ausgabe) ist wahr, wenn die Anzahl gleich der Voreinstellung ist.
- *F* - Überlauf (Ausgabe) ist wahr, wenn die Anzahl von 9999 auf 0 übertragen wird.

Die Aufwärts- und Abwärtszählkontakte sind flankenempfindlich, d. h. sie zählen nur, wenn der Kontakt von falsch auf wahr wechselt (oder von aus auf ein, wenn Sie das wünschen).

Der Bereich reicht von 0 bis 9999.

Es gibt auch Variablen für Zähler, die in Vergleichs- oder Operationsblöcken gelesen und/oder beschrieben werden können.

- %C`__xx__.D` - Zähler *xx* fertig (Boolesch, nur lesbar)
- %C`__xx__.E` - Zähler *xx* leerer Überlauf (boolesch, schreibgeschützt)
- %C`__xx__.F` - Zähler *xx* voller Überlauf (boolesch, nur Lesen)
- %C`__xx__.V` - Zähler *xx* aktueller Wert (Ganzzahl, Lesen oder Schreiben)
- %C`__xx__.P` - Zähler *xx* voreingestellt (Ganzzahl, lesen oder schreiben)

VERGLEICHEN

Für den arithmetischen Vergleich. Ist die Variable %XXX = zu dieser Zahl (oder der ausgewerteten Zahl)

Der Vergleichsblock wird wahr, wenn der Vergleich wahr ist. Sie können die meisten mathematischen Symbole verwenden:

- +, - , * , /, = (mathematische Standardzeichen)
- < (kleiner als), > (größer als), <= (kleiner oder gleich), >= (größer oder gleich), <> (ungleich)
- (,) trennen in Gruppen Beispiel %IF1=2,&%IF2<5 im Pseudocode bedeutet, wenn %IF1 gleich 2 ist und %IF2 kleiner als 5 ist, dann ist der Vergleich wahr. Beachten Sie das Komma, das die beiden Gruppen von Vergleichen trennt.
- ^ (Exponent), % (Modul), & (und), | (oder),. -
- ABS (absolut), MOY (französisch für Durchschnitt), AVG (Durchschnitt)

Zum Beispiel ABS(%W2)=1, MOY(%W1,%W2)<3.

In der Vergleichsgleichung sind keine Leerzeichen erlaubt. Zum Beispiel ist %C0.V>%C0.P ein gültiger Vergleichsausdruck, während %C0.V > %CO.P kein gültiger Ausdruck ist.

Unten auf der Seite befindet sich eine Liste von Variablen, die zum Lesen von und Schreiben in Kontaktplanobjekten verwendet werden können. Wenn ein neuer Vergleichsblock geöffnet wird, achten Sie darauf, das Symbol # zu löschen, wenn Sie einen Vergleich eingeben.

Um herauszufinden, ob die Wortvariable Nr. 1 kleiner als das 2-fache des aktuellen Wertes von Zähler Nr. 0 ist, lautet die Syntax wie folgt:

```
%W1<2*%C0.V
```

Um herauszufinden, ob S32in Bit 2 gleich 10 ist, würde die Syntax lauten:

```
%IW2=10
```

Hinweis: Compare verwendet das arithmetische Gleichheitszeichen und nicht das doppelte Gleichheitszeichen, an das Programmierer gewöhnt sind.

VARIABLENZUWEISUNG

Für die Variablenzuweisung, z. B. Zuweisung dieser Zahl (oder einer ausgewerteten Zahl) an diese Variable %xxx, gibt es zwei mathematische Funktionen MINI und MAXI, die eine Variable auf Maximal- (0x80000000) und Minimalwerte (0x07FFFFFF) prüfen (man denke an vorzeichenbehaftete Werte) und verhindern, dass diese überschritten werden.

Wenn ein neuer Variablenzuweisungsblock geöffnet wird, achten Sie darauf, das Symbol # zu löschen, wenn Sie eine Zuweisung eingeben.

Um der Timer-Voreinstellung von IEC Timer 0 den Wert 10 zuzuweisen, lautet die Syntax:

```
%TM0.P=10
```

Um den Wert von 12 auf s32out Bit 3 zuzuweisen, wäre folgendes:

```
%QW3=12
```

NOTE Wenn Sie einer Variablen mit dem Variablenzuweisungsblock einen Wert zuweisen, bleibt der Wert erhalten, bis Sie mit dem Variablenzuweisungsblock einen neuen Wert zuweisen. Der zuletzt zugewiesene Wert wird wiederhergestellt, wenn LinuxCNC gestartet wird.

Die folgende Abbildung zeigt ein Zuweisungs- und ein Vergleichsbeispiel. %QW0 ist ein S32out-Bit und %IW0 ist ein S32in-Bit. In diesem Fall wird der HAL-Pin `classicladder.0.s32out-00` auf einen Wert von 5 gesetzt, und wenn der HAL-Pin `classicladder.0.s32in-00` 0 ist, wird der HAL-Pin `classicladder.0.out-00` auf True gesetzt.

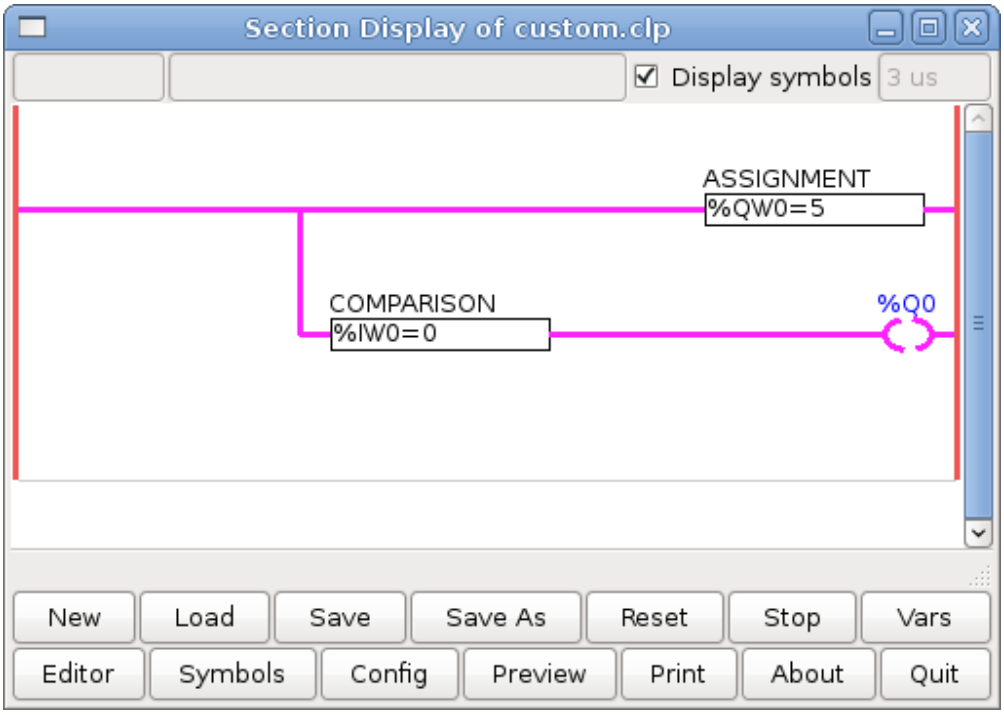


Figure 117. Beispiel für Zuordnen/Vergleichen von mit SPS



Figure 118. Beispiel für einen Zuweisungsausdruck



Figure 119. Beispiel für einen Vergleichsausdruck

SPULEN (engl. coils)

Spulen stellen Relaisspulen dar. Sie werden durch den ihnen zugewiesenen variablen Buchstaben und die Nummer gesteuert.

Der variable Buchstabe kann B oder Q sein und die Nummer kann bis zu einer dreistelligen Zahl sein, z. B. %Q3 oder %B123. Q-Spulen steuern HAL-Ausgangsstifte, z. B. wenn %Q15 aktiviert ist, wird der HAL-Pin `classicladder.0.out-15` wahr. B-Spulen sind interne Spulen, die zur Steuerung des

Programmablaufs verwendet werden.

- N.O. COIL' - Eine Relaisspule: Wenn die Spule erregt ist, wird ihr Kontakt, der normalerweise offen ist (kurz: N.O.), geschlossen (eingeschaltet, wahr, etc.) und der Strom kann passieren.
- N.C. COIL' - Eine Relaisspule, die ihre Kontakte umkehrt: Wenn die Spule erregt wird, dann wird der normalerweise geschlossene Kontakt (engl. kurz: N.C.) geöffnet (ausgeschaltet, falsch, usw.) und der Stromfluss wird unterbrochen.
- *SET COIL* - Eine Relaisspule mit verriegelten Kontakten: Wenn die Spule erregt ist, wird der Kontakt geschlossen und bleibt dies.
- *RESET COIL* - (eine Relaisspule mit selbsthaltenden Kontakten) Wenn die Spule erregt ist, wird der N.0.-Kontakt offen gehalten.
- *JUMP COIL* - (eine *goto* Spule), wenn die Spule erregt wird, springt das Kontaktplanprogramm zu einem Strompfad (im Abschnitt CURRENT) - die Sprungpunkte werden durch eine Strompfadbezeichnung gekennzeichnet. (Sprossenbeschriftungen in der Abschnittsanzeige, oben links, hinzufügen.)
- *CALL COIL* - Eine *gosub*-Spule: Wenn die Spule erregt wird, dann springt das Programm zu einem Unterprogrammabschnitt, der durch eine Unterprogrammnummer gekennzeichnet ist - Unterprogramme werden mit SR0 bis SR9 bezeichnet (bestimmen Sie sie im Abschnittsmanager).

WARNING	Wenn Sie einen NC-Kontakt mit einer NC-Spule verwenden, funktioniert die Logik (wenn die Spule erregt ist, wird der Kontakt geschlossen), aber das ist wirklich schwer zu verstehen!

SPRUNGSPULE (engl. jump coil)

Eine JUMP COIL wird verwendet, um zu einem anderen Abschnitt zu *springen*, wie ein goto in der Programmiersprache BASIC.

Oben links im Fenster der Abschnittsanzeige sehen Sie ein kleines Beschriftungsfeld und ein längeres Kommentarfeld daneben. Gehen Sie nun auf Editor→Ändern und dann zurück zu dem kleinen Kästchen, um einen Namen einzugeben.

Fügen Sie einfach einen Kommentar im Kommentarbereich hinzu. Diese Bezeichnung ist nur der Name dieser Sprosse und wird von der JUMP COIL verwendet, um zu erkennen, wohin sie gehen soll.

Wenn Sie eine JUMP COIL platzieren, fügen Sie sie an der äußersten rechten Position ein und ändern die Beschriftung in die Sprosse, zu der Sie JUMPEN wollen.

RUFSPULE (engl. call coil)

Eine CALL COIL wird verwendet, um zu einem Unterprogramm zu gelangen und dann zurückzukehren, wie ein gosub in der Programmiersprache BASIC.

Gehen Sie zum Fenster Abschnittsmanager und klicken Sie auf die Schaltfläche Abschnitt hinzufügen. Sie können diesen Abschnitt benennen, die Sprache (Kontaktplan oder sequentiell) und den Typ (Haupt- oder Unterprogramm) auswählen.

Wählen Sie eine Unterprogrammnummer (z. B. SR0). Es wird ein leerer Abschnitt angezeigt und Sie können Ihr Unterprogramm erstellen.

Wenn Sie das getan haben, gehen Sie zurück zum Abschnittsmanager und klicken Sie auf Ihren Hauptabschnitt (Standardname prog1).

Jetzt können Sie eine CALL COIL in Ihr Programm einfügen. CALL COILs sind an der äußersten rechten Position im Strompfad zu platzieren.

Vergessen Sie nicht, die Beschriftung in die Nummer des Unterprogramms zu ändern, die Sie zuvor gewählt haben.

8.2.7. ClassicLadder Variablen

Diese Variablen werden in COMPARE oder OPERATE verwendet, um Informationen über Kontaktplanobjekte zu erhalten oder deren Spezifikationen zu ändern, z. B. um einen Zähler zu ändern oder um zu sehen, ob ein Timer fertig ist.

Liste der Variablen :

- `%B`*xxx* - Bitspeicher *xxx* (boolesch)
- `%W`*xxx* - Word-Speicher *xxx* (32 Bit Vorzeichen)
- `%IW`*xxx* - Word memory *xxx* (Sint in pin)
- `%QW`*xxx* - Word memory *xxx* (Sint out pin)
- `%IF`*xx* - Word memory *xx* (Real in pin) (**converted to Sint in ClassicLadder**)
- `%QF`*xx* - Word memory *xx* (Real out pin) (**converted to Sint in ClassicLadder**)
- `%T`__xx__.R` - Timer *xx* läuft (boolesch, schreibgeschützt für Benutzer)
- `%T`__xx__.D` - Timer *xx* erledigt (Boolescher Wert, nur für Benutzer)
- `%T`__xx__.V` - Timer *xx* aktueller Wert (ganze Zahl, schreibgeschützt für Benutzer)
- `%T`__xx__.P` - Timer *xx* Voreinstellung (ganze Zahl)
- `%TM`__xxx__.Q` - Timer *xxx* fertig (Boolesch, lesen/schreiben)
- `%TM`__xxx__.P` - Timer *xxx* Voreinstellung (ganze Zahl, Schreiblese)
- `%TM`__xxx__.V` - Timer *xxx* Wert (ganze Zahl, lesender Schreibzugriff)
- `'%M`__xx__.R` - Monostabile *xx* läuft (boolesch)
- `%M`__xx__.V` - Monostabile *xx* aktueller Wert (Ganzzahl, nur vom Benutzer lesbar)
- `%M`__xx__.P` - Monostabile *xx* Voreinstellung (Ganzzahl)
- `%C`__xx__.D` - Zähler *xx* fertig (Boolesch, Benutzer schreibgeschützt)
- `%C`__xx__.E` - Zähler *xx* leerer Überlauf (Boolean, nur vom Benutzer gelesen)
- `%C`__xx__.F` - Zähler *xx* voller Überlauf (Boolean, schreibgeschützter Benutzer)
- `%C`__xx__.V` - Zähler *xx* aktueller Wert (Ganzzahl)

- %C`__xx__.P` - Zähler *xx* Voreinstellung (ganze Zahl)
- `%I`*xxx* - Physical input *xxx* (Boolean) (HAL input bool)
- `%Q`*xxx* - Physical output *xxx* (Boolean) (HAL output bool)
- `%X`*xxx* - Aktivität von Schritt *xxx* (sequentielle Sprache)
- %X`__xxx__.V` - Zeit der Aktivität in Sekunden von Schritt *xxx* (sequenzielle Sprache)
- `%E`*xx* - Fehler (Boolean, read write(wird überschrieben))
- *Indizierte oder vektorisierte Variablen* - Dies sind Variablen, die durch eine andere Variable indiziert werden. Einige mögen dies vektorisierte Variablen nennen. Beispiel: %W0[%W4] ⇒ Wenn %W4 gleich 23 ist, entspricht es %W23

8.2.8. GRAFCET (State Machine) Programmierung

WARNING

Dies ist wahrscheinlich die am wenigsten genutzte und am schlechtesten verstandene Funktion von ClassicLadder. Die Ablaufprogrammierung wird verwendet, um sicherzustellen, dass eine Reihe von Kontaktplanereignissen immer in einer bestimmten Reihenfolge abläuft. Ablaufprogramme funktionieren nicht allein. Es gibt immer auch ein Kontaktplanprogramm, das die Variablen steuert. Hier sind die Grundregeln für Ablaufprogramme:

- Regel 1: Ausgangssituation - Die Ausgangssituation wird durch die Anfangsschritte charakterisiert, die sich zu Beginn des Vorgangs per Definition im aktiven Zustand befinden; es muss mindestens einen Anfangsschritt geben.
- Regel 2 : R2, Löschung einer Transition - Eine Transition ist entweder aktiviert oder deaktiviert. Sie gilt als aktiviert, wenn alle unmittelbar vorangehenden Schritte, die mit dem entsprechenden Übergangssymbol verbunden sind, aktiv sind, ansonsten ist sie deaktiviert. Eine Transition kann nur gelöscht werden, wenn sie aktiviert ist und die zugehörige Transitionsbedingung wahr ist.
- Regel 3 : R3, Entwicklung aktiver Schritte - Die Löschung eines Übergangs führt gleichzeitig zum aktiven Zustand des/der unmittelbar folgenden Schrittes/Schritte und zum inaktiven Zustand des/der unmittelbar vorangehenden Schrittes/Schritte.
- Regel 4 : R4, Gleichzeitige Löschung von Übergängen - Alle gleichzeitig gelöschten Übergänge werden gleichzeitig gelöscht.
- Regel 5 : R5, Gleichzeitiges Aktivieren und Deaktivieren eines Schrittes - Wenn während des Betriebs ein Schritt gleichzeitig aktiviert und deaktiviert wird, hat die Aktivierung Vorrang.

Dies ist das Fenster des SEQUENTIAL-Editors. (Beginnend von oben links):
Auswahlpfeil, Radiergummi
Gewöhnlicher Schritt, Anfangsschritt (Start)
Transition, Schritt und Transition
Übergang Link-unten, Übergang Link-oben
Durchgang Link-unten, Durchgang Link-oben Sprung
Link, Kommentarfeld

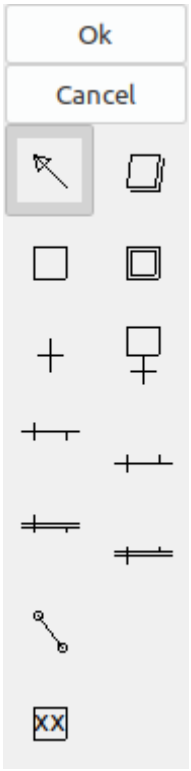


Figure 120. Fenster des Sequenzeditors

- *ORDINARY STEP* - hat für jeden eine eindeutige Nummer
- *STARTING STEP* - ein sequentielles Programm muss einen haben. Hier beginnt das Programm.
- *TRANSITION* - zeigt die Variable an, die wahr sein muss, damit die Steuerung zum nächsten Schritt übergeht.
- *STEP AND TRANSITION* - der Einfachheit halber kombiniert
- *TRANSITION LINK-DOWNSIDE* - teilt den Logikfluss in eine von zwei möglichen Linien auf, je nachdem, welcher der nächsten Schritte zuerst wahr ist (denken Sie an die ODER-Logik)
- *TRANSITION LINK=UPSIDE* - verbindet zwei (ODER) Logiklinien wieder zu einer
- *PASS-THROUGH-LINK-DOWNSIDE* - teilt den Logikfluss in zwei Zeilen auf, dass BEIDE wahr sein müssen, um fortzufahren (Think AND logic)
- *PASS-THROUGH-LINK-UPSIDE* - kombiniert zwei gleichzeitige (UND logische) Logiklinien wieder zusammen
- *JUMP LINK* - verbindet Schritte, die nicht untereinander liegen, z. B. das Verbinden des letzten Schritts mit dem ersten
- *COMMENT BOX* - wird verwendet, um Kommentare hinzuzufügen

Um Verknüpfungen zu verwenden, müssen Sie bereits Schritte platziert haben. Wählen Sie die Art der Verknüpfung und dann die beiden Schritte oder Transaktionen nacheinander aus. Das erfordert Übung!

Bei sequentieller Programmierung: Die Variable `%X`__xxx__ (z. B. `%X5´) wird verwendet, um festzustellen, ob ein Schritt aktiv ist. Die Variable `%X`__xxx__.V` (z. B. `%X5.V`) wird verwendet, um festzustellen, wie lange der Schritt aktiv war. Die Variablen %X und %X.v werden in der LADDER-Logik verwendet. Die den Transitionen zugeordneten Variablen (z. B. %B) steuern, ob die Logik zum nächsten Schritt übergeht. Nachdem ein Schritt aktiv geworden ist, hat die Übergangsvariable, die

ihn aktiv werden ließ, keine Kontrolle mehr über ihn. Der letzte Schritt muss nur noch zum Anfangsschritt zurückspringen (JUMP LINK).

8.2.9. Modbus

Zu beachtende Punkte:

- Modbus ist ein Nicht-Echtzeitprogramm, so dass es auf einem stark belasteten Computer zu Latenzproblemen kommen kann.
- Modbus eignet sich nicht wirklich für harte Echtzeit-Ereignisse wie die Positionssteuerung von Motoren oder die Steuerung von Notausschaltern.
- Das ClassicLadder GUI muss ausgeführt werden, damit Modbus ausgeführt werden kann.
- Modbus ist noch nicht ganz fertig, so dass nicht alle Modbus-Funktionen zur Verfügung stehen.

Um MODBUS zu initialisieren, müssen Sie dies beim Laden des ClassicLadder Nicht-Echtzeitprogramms angeben.

Laden von Modbus

```
loadusr -w classicladder --modmaster myprogram.clp
```

Das `-w` bewirkt, dass HAL wartet, bis Sie ClassicLadder schließen, bevor es die Echtzeit-Sitzung beendet. ClassicLadder lädt auch einen TCP-Modbus-Slave, wenn Sie `--modserver` auf der Kommandozeile hinzufügen.

Modbus-Funktionen

- *1* - Spulen lesen
- *2* - Eingänge lesen
- *3* - Halteregister lesen
- *4* - Eingaberegister lesen
- *5* - einzelne Spulen schreiben
- *6* - Einzelnes Register schreiben
- *8* - Echo-Test
- *15* - mehrere Spulen schreiben
- *16* - mehrere Register schreiben

Wenn Sie beim Laden des Nicht-Echtzeit-Programms ClassicLadder kein *--modmaster* angeben, wird diese Seite nicht angezeigt.

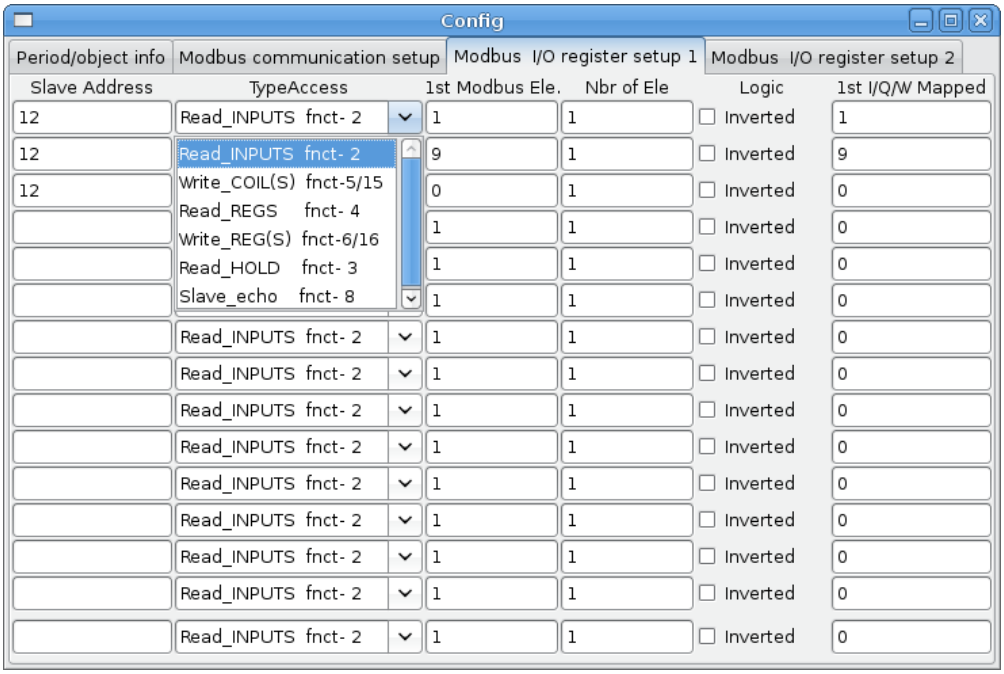


Figure 121. Modbus I/O-Konfiguration

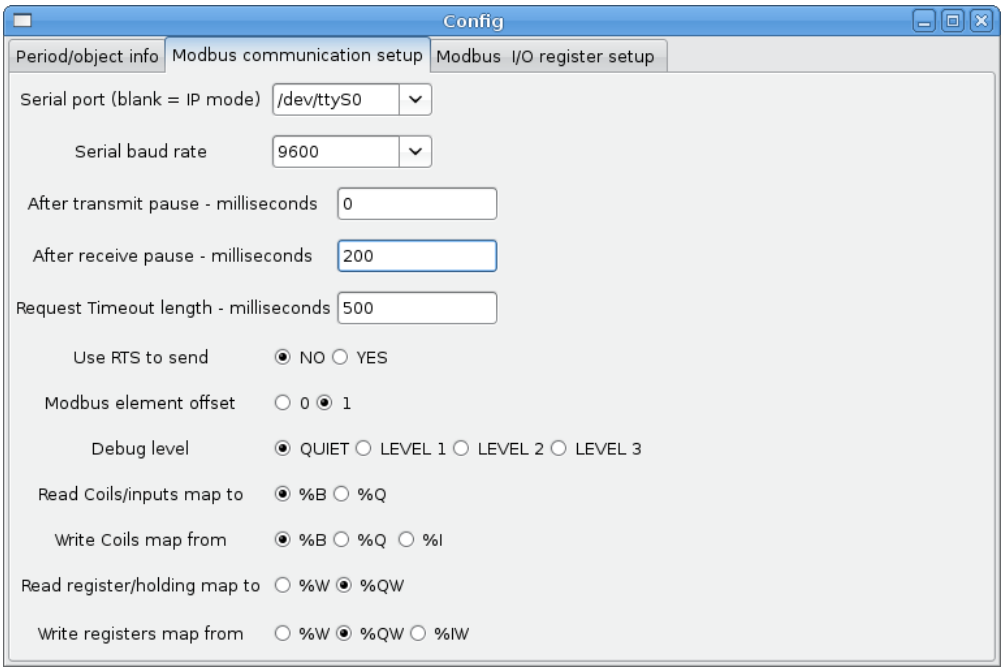


Figure 122. Modbus-Kommunikationskonfiguration

- *SERIAL PORT* - Für IP leer. Für seriell der Ort/Name des seriellen Treibers, z.B. /dev/ttyS0 (oder /dev/ttyUSB0 für einen USB-zu-Seriell-Konverter).
- *SERIAL SPEED* - Sollte auf Geschwindigkeit eingestellt sein, für die der Slave eingestellt ist - 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 werden unterstützt.
- *PAUSE AFTER TRANSMIT* - Pause (Millisekunden) nach dem Senden und vor dem Empfang der Antwort, einige Geräte benötigen mehr Zeit (z. B. USB-zu-Seriell-Konverter).
- *PAUSE INTER-FRAME* - Pause (Millisekunden) nach Erhalt der Antwort vom Slave. Dadurch wird der Arbeitszyklus von Anforderungen festgelegt (es ist eine Pause für JEDE Anforderung).
- *REQUEST TIMEOUT LENGTH* - Länge (Millisekunden) der Zeit, bevor wir entscheiden, dass der Slave nicht geantwortet hat.

- *MODBUS ELEMENT OFFSET* - wird verwendet, um die Elementnummern um 1 zu kompensieren (für Hersteller, die Unterschiede nummerieren).
- *DEBUG LEVEL* - Setzen Sie dies auf 0-3 (0, um das Drucken von Debug-Informationen neben No-Response-Fehlern zu stoppen).
- *READ COILS/INPUTS MAP TO* - Wählen Sie, welche Variablen die gelesenen Spulen/Eingänge aktualisieren sollen. (B oder Q).
- *WRITE COILS MAP TO* - Wählen Sie aus, von welchen Variablen, von denen Schreibspulen aktualisiert werden sollen (B, Q oder I).
- *READ REGISTERS/HOLDING* - Wählen Sie aus, welche Variablen durch das Lesen von Registern aktualisiert werden sollen (W oder QW).
- *WRITE REGISTERS MAP TO* - Wählen Sie aus, von welchen Variablen die Leseregister aktualisiert werden (W, QW oder IW).
- *SLAVE-ADRESSE* - Für serielle die Slave-ID-Nummer normalerweise auf dem Slave-Gerät einstellbar (normalerweise 1-256). Für IP die Slave-IP-Adresse plus optional die Portnummer.
- *TYPE ACCESS* - Dies wählt den MODBUS-Funktionscode aus, der an den Slave gesendet werden soll (z. B. welche Art von Anfrage).
- *COILS / INPUTS* - Eingänge und Spulen (Bits) werden aus I-, B- oder Q-Variablen gelesen / geschrieben (Benutzerauswahl).
- *REGISTERS (WORDS)* - Register (Wörter/Zahlen) werden IW-, W- oder QW-Variablen zugeordnet (Benutzerauswahl).
- *1st MODBUS ELEMENT* - Die Adresse (oder Registernummer) des ersten Elements in einer Gruppe (Denken Sie daran, MODBUS ELEMENT OFFSET richtig einzustellen).
- *ANZAHL DER ELEMENTE* - Die Anzahl der Elemente in dieser Gruppe.
- *LOGIC* - Sie können die Logik hier umkehren.
- *1st%I%Q IQ WQ MAPPED* - Dies ist die Startnummer von %B, %I, %Q, %W, %IW oder %QW Variablen, die auf/von der Modbus-Elementgruppe zugeordnet werden (beginnend mit der ersten Modbus-Elementnummer).

Im obigen Beispiel: Portnummer - für meinen Computer war /dev/ttyS0 meine serielle Schnittstelle.

Die serielle Geschwindigkeit ist auf 9600 Baud eingestellt.

Die Slave-Adresse ist auf 12 gesetzt (auf meinem VFD kann ich dies von 1-31 einstellen, was bedeutet, dass ich auf einem System maximal mit 31 VFDs sprechen kann).

Die erste Zeile ist für 8 Eingangsbits eingerichtet, beginnend mit der ersten Registernummer (Register 1). Die Registernummern 1-8 werden also auf die %B-Variablen von ClassicLadder abgebildet, beginnend bei %B1 und endend bei %B8.

Die zweite Zeile ist für 2 Ausgangsbits ab der neunten Registernummer (Register 9) eingestellt, so dass die Registernummern 9-10 auf die %Q-Variablen von ClassicLadder abgebildet werden, die bei %Q9 beginnen und bei %Q10 enden.

Die dritte Zeile ist so eingestellt, dass 2 Register (je 16 Bit) geschrieben werden, beginnend mit der 0ten Registernummer (Register 0), so dass die Registernummern 0-1 auf die %W-Variablen des ClassicLadder abgebildet werden, beginnend mit %W0 und endend mit %W1.

Es ist leicht, einen Off-by-One-Fehler zu machen, da die Modbus-Elemente manchmal bei 1 und nicht bei 0 referenziert werden (eigentlich ist das laut Standard so vorgesehen!). Sie können die Optionsschaltfläche für den Modbus-Element-Offset verwenden, um dies zu vermeiden.

In den Unterlagen zu Ihrem Modbus-Slave-Gerät finden Sie Informationen darüber, wie die Register aufgebaut sind - es gibt keine Standardmethode.

Die Parameter SERIAL PORT, PORT SPEED, PAUSE und DEBUG-Level können geändert werden (beim Schließen des Konfigurationsfensters werden die Werte übernommen, die Radio-Buttons gelten jedoch sofort).

Um die Echo-Funktion zu verwenden, wählen Sie die Echo-Funktion aus und fügen Sie die Slave-Nummer hinzu, die Sie testen möchten. Sie müssen keine Variablen angeben.

Die Nummer 257 wird an die von Ihnen angegebene Slave-Nummer gesendet und der Slave sollte sie zurücksenden. Sie müssen ClassicLadder in einem Terminal laufen lassen, um die Nachricht zu sehen.

8.2.10. MODBUS-Einstellungen

Seriell:

- ClassicLadder verwendet das RTU-Protokoll (nicht ASCII).
- 8 Datenbits, keine Parität und 1 Stoppbit werden auch als 8-N-1 bezeichnet.
- Die Baudrate muss für Slave und Master gleich sein. ClassicLadder kann nur eine Baudrate haben, also müssen alle Slaves auf die gleiche Rate eingestellt werden.
- Das Pausenintervall ist die Zeitspanne, die nach dem Empfang einer Antwort pausiert wird.
- MODBUS_TIME_AFTER_TRANSMIT ist die Länge der Pause nach dem Senden einer Anfrage und vor dem Empfang einer Antwort (dies hilft offenbar bei langsamen USB-Wandlern).

MODBUS-Info

- ClassicLadder kann verteilte Eingänge/Ausgänge auf Modulen verwenden, die das Modbus-Protokoll verwenden ("Master": abfragende Slaves).
- Die Slaves und ihre I/O können im Konfigurationsfenster konfiguriert werden.
- Es sind 2 exklusive Modi verfügbar: Ethernet mit Modbus/TCP und seriell mit Modbus/RTU.
- Es wird keine Parität verwendet.
- Wenn kein Portname für die serielle Schnittstelle eingestellt ist, wird der TCP/IP-Modus verwendet...
- Die Slave-Adresse ist die Slave-Adresse (Modbus/RTU) oder die IP-Adresse.
- Die IP-Adresse kann durch die zu verwendende Portnummer ergänzt werden (xx.xx.xx.xx:pppp), andernfalls wird standardmäßig der Port 9502 verwendet.

- Für die Tests wurden 2 Produkte verwendet: ein Modbus/TCP-Produkt (Adam-6051, https://www.advantech.com) und ein serielles Modbus/RTU-Produkt (https://www.ipac.ws).
- Siehe Beispiele: adam-6051 und modbus_rtu_serial.
- Weblinks: https://www.modbus.org und dieser interessante Link: https://www.iatips.com/modbus.html
- MODBUS TCP SERVER ENTHALTEN
- ClassicLadder hat einen Modbus/TCP-Server integriert. Der Standard-Port ist 9502. (der vorherige Standard 502 erfordert, dass die Anwendung mit Root-Rechten gestartet werden muss).
- Die Liste der unterstützten Modbus-Funktionscodes lautet: 1, 2, 3, 4, 5, 6, 15 und 16.
- Die Korrespondenztabelle der Modbus-Bits und -Worte ist eigentlich nicht parametrisch und entspricht direkt den Variablen %B und %W.

Weitere Informationen zum Modbus-Protokoll finden Sie im Internet.

https://www.modbus.org/

Kommunikationsfehler

Wenn ein Kommunikationsfehler auftritt, wird ein Warnfenster angezeigt (wenn die grafische Benutzeroberfläche läuft) und %E0 ist wahr. Modbus wird weiterhin versuchen, zu kommunizieren. Der %E0-Wert kann verwendet werden, um eine Entscheidung auf der Grundlage des Fehlers zu treffen. Ein Timer könnte verwendet werden, um die Maschine anzuhalten, wenn die Zeit abgelaufen ist, usw.

8.2.11. Fehlersuche bei Modbus-Problemen

Eine gute Referenz für das Protokoll: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. Wenn Sie linuxcnc/classicladder von einem Terminal aus starten, werden die Modbus-Befehle und Slave-Antworten ausgegeben.

Hier wird der ClassicLadder so eingestellt, dass er den Slave 1 auffordert, die Holding-Register (Funktionscode 3) ab Adresse 8448 (0x2100) zu lesen. Wir fordern die Rückgabe von 1 (2 Byte breiten) Datenelement an. Wir ordnen es einer ClassicLadder-Variablen zu, startend bei 2.

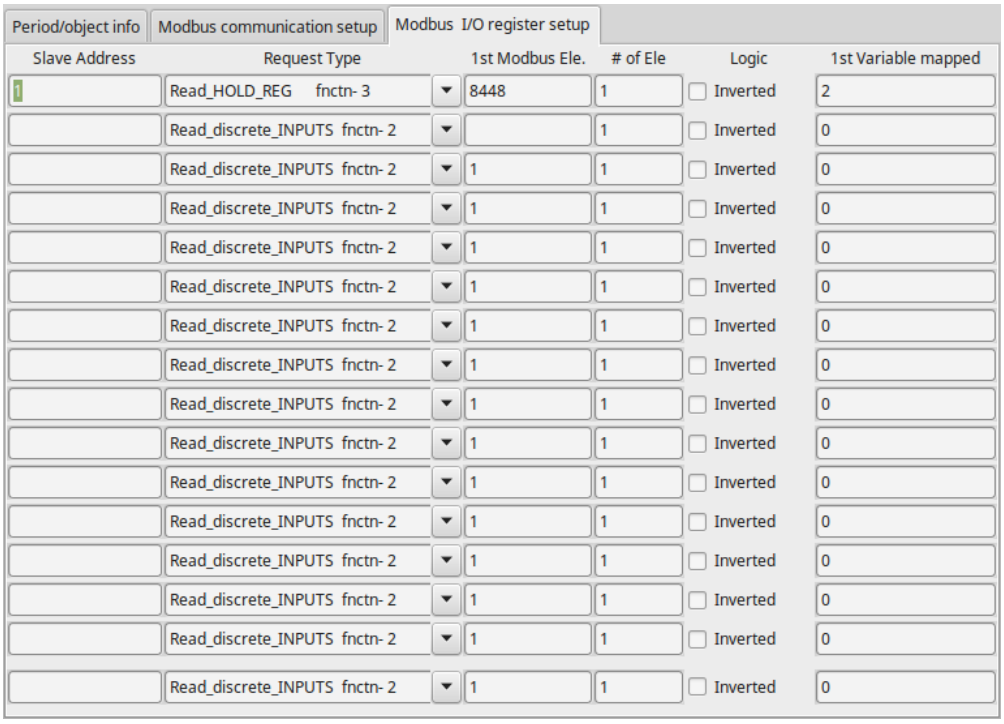


Figure 123. Modbus I/O-Register-Setup

Hinweis in diesem Bild haben wir die Debug-Ebene auf 1 gesetzt, so dass Modbus-Nachrichten an das Terminal ausgegeben werden. Wir haben unsere Lese- und Schreibregister den %W-Variablen von ClassicLadder zugeordnet, so dass unsere zurückgegebenen Daten in %W2 sind, wie in dem anderen Bild, in dem wir die Daten ab dem 2. Element zugeordnet haben.

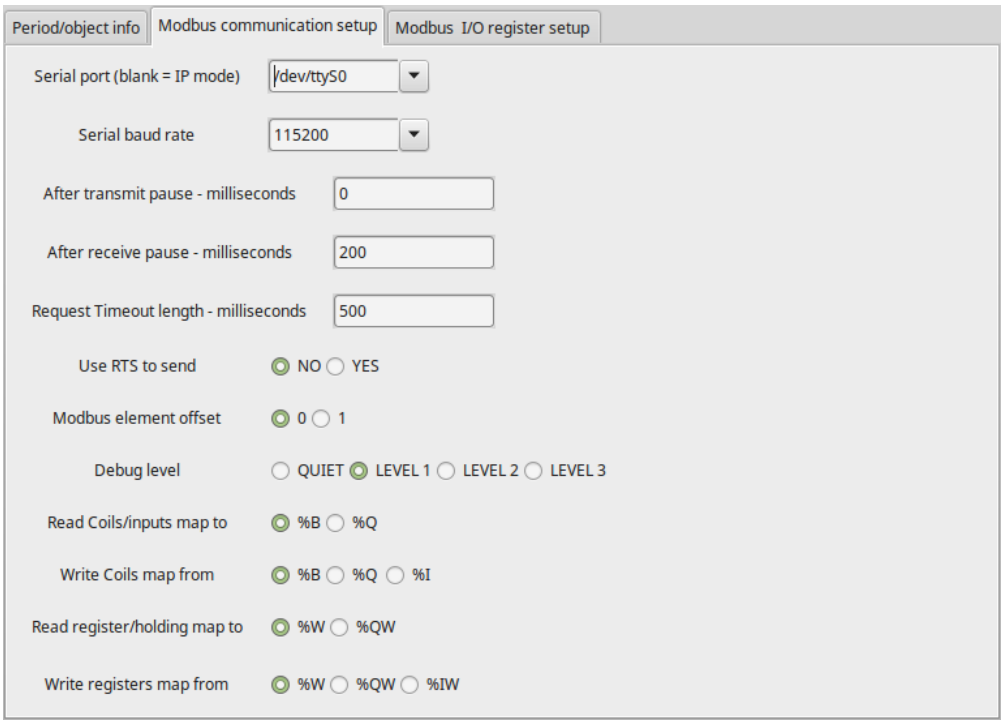


Figure 124. Einrichtung der Modbus-Kommunikation

Anfrage

Betrachten wir ein Beispiel für das Lesen eines Hold-Registers bei 8448 Decimal (0x2100 Hex).

Blick in die Modbus-Protokollreferenz:

Table 42. Halteregisteranforderung lesen

Name	Anzahl Bytes	Wert (hex)
Funktionscode	(1 Byte)	3 (0x03)
Startadresse	(2 Bytes)	0 - 65535 (0x0000 bis 0xFFFF)
Anzahl von Registern	(2 Bytes)	1 bis 125 (0x7D)
Prüfsumme	(2 Bytes)	Automatisch berechnet

Hier ist ein Beispiel für einen gesendeten Befehl, wie er im Terminal ausgedruckt wird (alles in Hex):

```
INFO CLASSICLADDER-   Modbus I/O module to send: Lgt=8 <-  Slave address-1  Function
code-3  Data-21 0 0 1 8E 36
```

Bedeutung (Hex):

- Lgt = 8 = Nachricht ist 8 Bytes lang, einschließlich Slave-Nummer und Prüfsummen-Nummer
- Slave-Nummer = 1 (0x1) = Slave-Adresse 1
- Funktionscode = 3 (0x3) = Halteregister lesen
- Start bei Adresse = Highbyte 33 (0x21) Lowbyte 0 (0x00) = kombinierte Adresse = 8448 (0x2100)
- Anzahl der Register = 1 (0x1) = 1 2-Byte-Register zurückgeben (Halte- und Leseregister sind immer 2 Byte breit)
- Prüfsumme = Highbyte 0x8E Lowbyte 0x36 = (0x8E36)

Fehlerreaktion

Bei einer Fehlerantwort sendet er den Funktionscode plus 0x80, einen Fehlercode und eine Prüfsumme. Eine Fehlerantwort zu erhalten bedeutet, dass der Slave den Anforderungsbefehl sieht, aber keine gültigen Daten liefern kann. Schauen Sie in der Modbus-Protokollreferenz nach:

Table 43. Fehler bei Funktionscode 3 (Lesen des Holdingregisters)

Name	Anzahl Bytes	Wert (hex)
Fehlercode	1 Byte	131 (0x83)
Ausnahmecode	1 Byte	1-4 (0x01 bis 0x04)
Prüfsumme	(2 Bytes)	Automatisch berechnet

Bedeutung des Ausnahmecodes:

- 1 - illegal Funktion
- 2 - unzulässige Datenadresse
- 3 - unzulässiger Datenwert
- 4 - Ausfall des Slave-Geräts

Hier ist ein Beispiel für einen empfangenen Befehl, wie er im Terminal ausgedruckt wird (alles in Hex):

```
INFO CLASSICLADDER-   Modbus I/O module received: Lgt=5 ->   (Slave address-1  Function
code-83 ) 2 C0 F1
```

Bedeutung (Hex):

- Slave-Nummer = 1 (0x1) = Slave-Adresse 1
- Funktionscode = 131 (0x83) = Fehler beim Lesen des Holdingregisters
- Fehlercode = 2 (0x2) = unzulässige Datenadresse angefordert
- Prüfsumme = (0x8E36)

Datenantwort

Blick auf das Protokoll über die Antwort:

Table 44. Datenantwort für Funktionscode 3 (Lesen Holdingregister)

Name	Anzahl Bytes	Wert (hex)
Funktionscode	1 Byte	3 (0x03)

Name	Anzahl Bytes	Wert (hex)
Anzahl der Bytes	1 Byte	2 x N*
Wert des Registers	N* x 2 Bytes	Rückgabewert der angeforderten Adresse
Prüfsumme	(2 Bytes)	automatisch berechnet

*N = Anzahl der Register

Hier ist ein Beispiel für einen empfangenen Befehl, wie er im Terminal ausgedruckt wird (alles in Hex):

```
INFO CLASSICLADDER-   Modbus I/O module received: Lgt=7 ->   (Slave address-1  Function
code-3 2 0 0 B8 44)
```

Bedeutung (Hex):

- Slave-Nummer = 1 (0x1) = Slave-Adresse 1
- Angeforderter Funktionscode = 3 (0x3) = Lesen des Holdingregisters angefordert
- Anzahl der Byte-Register = 2 (0x1) = Rückgabe von 2 Bytes (jeder Registerwert ist 2 Bytes breit)
- Wert des Highbytes = 0 (0x0) = Highbyte-Wert der Adresse 8448 (0x2100)
- Wert des Lowbyte = 0 (0x0) = Wert des Highbyte der Adresse 8448 (0x2100)
- Prüfsumme = (0xB844)

(High- und Low-Bytes werden zu einem 16-Bit-Wert kombiniert und dann an die ClassicLadder-Variable übertragen). Leseregister können auf %W oder %QW (interner Speicher oder HAL-Out-Pins) abgebildet werden. Schreibregister können auf %W, %QW oder %IW (interner Speicher, HAL-Out-Pins oder HAL-In-Pins) abgebildet werden: Wenn mehrere Register in einem Lese-/Schreibvorgang angefordert werden, sind die Variablennummern nach der ersten fortlaufend.

MODBUS-Fehler

- In Vergleichsblöcken wird die Funktion %W=ABS(%W1-%W2) akzeptiert, aber nicht korrekt berechnet. Nur %W0=ABS(%W1) ist derzeit zulässig.
- Wenn Sie ein Kontaktplanprogramm laden, werden Modbus-Informationen geladen, aber ClassicLadder wird nicht angewiesen, Modbus zu initialisieren. Sie müssen Modbus initialisieren, wenn Sie die GUI zum ersten Mal laden, indem Sie *--modmaster* hinzufügen.
- Wenn der Abschnittsmanager über der Abschnittsanzeige platziert wird, über die Bildlaufleiste hinweg, und auf Beenden geklickt wird, stürzt das Nicht-Echtzeit-Programm ab.

- Wenn Sie *--modmaster* verwenden, müssen Sie gleichzeitig das Kontaktplanprogramm laden, sonst funktioniert nur TCP.
- das Lesen/Schreiben mehrerer Register im Modbus weist Prüfsummenfehler auf.

8.2.12. Einrichten von ClassicLadder

In diesem Abschnitt werden die Schritte beschrieben, die erforderlich sind, um ClassicLadder zu einer vom StepConf Wizard generierten Konfiguration hinzuzufügen. Auf der Seite "Erweiterte Konfigurationsoptionen" des StepConf-Assistenten aktivieren Sie "Classic Ladder PLC einbeziehen".

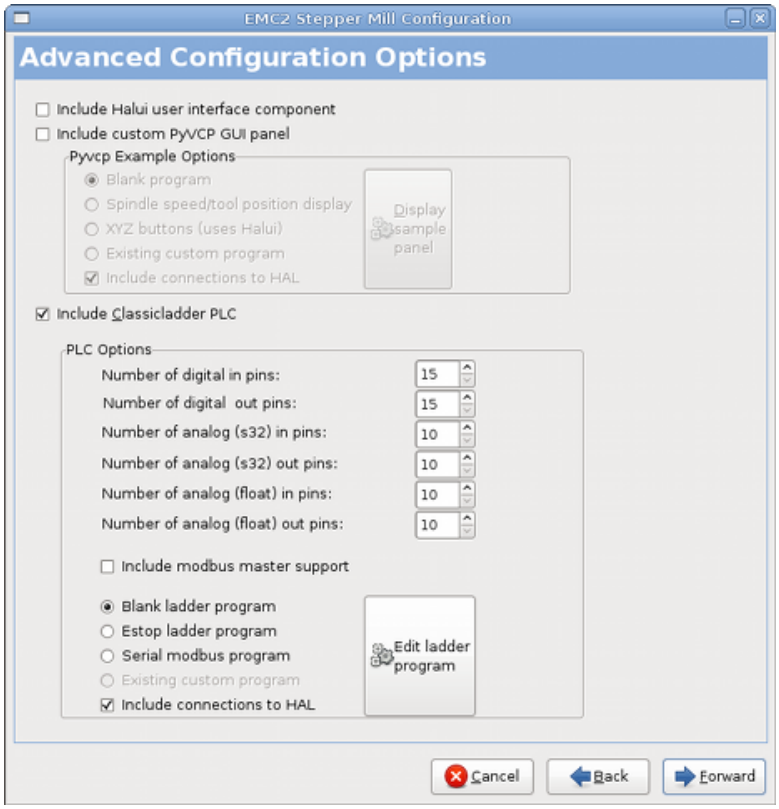


Figure 125. StepConf ClassicLadder

Hinzufügen der Module

Wenn Sie den StepConf-Assistenten zum Hinzufügen von ClassicLadder verwendet haben, können Sie diesen Schritt überspringen.

Um ClassicLadder manuell hinzuzufügen, müssen Sie zunächst die Module hinzufügen. Dazu fügen Sie der Datei custom.hal ein paar Zeilen hinzu.

Diese Zeile lädt das Echtzeitmodul:

```
loadrt classicladder_rt
```

Diese Zeile fügt dem Servo-Thread die Funktion ClassicLadder hinzu:

```
addf classicladder.0.refresh servo-thread
```

Hinzufügen der Kontaktplanlogik

Starten Sie nun Ihre Konfiguration und wählen Sie "Datei/Kontaktplan-Editor", um die GUI des klassischen Kontaktplans zu öffnen. Sie sollten ein leeres Fenster für die Abschnittsanzeige und den Abschnittsmanager sehen, wie oben gezeigt. Im Fenster Abschnittsanzeige öffnen Sie den Editor. Wählen Sie im Editor-Fenster "Ändern". Jetzt erscheint ein Fenster Eigenschaften und die Abschnittsanzeige zeigt ein Raster an. Das Gitter ist eine Sprosse des Leiters. Die Sprosse kann Verzweigungen enthalten. Eine einfache Sprosse hat einen Eingang, eine Verbindungslinie und einen Ausgang. Eine Sprosse kann bis zu sechs horizontale Zweige haben. Es ist zwar möglich, mehr als einen Stromkreis in einer Sprosse zu haben, aber die Ergebnisse sind nicht vorhersehbar.

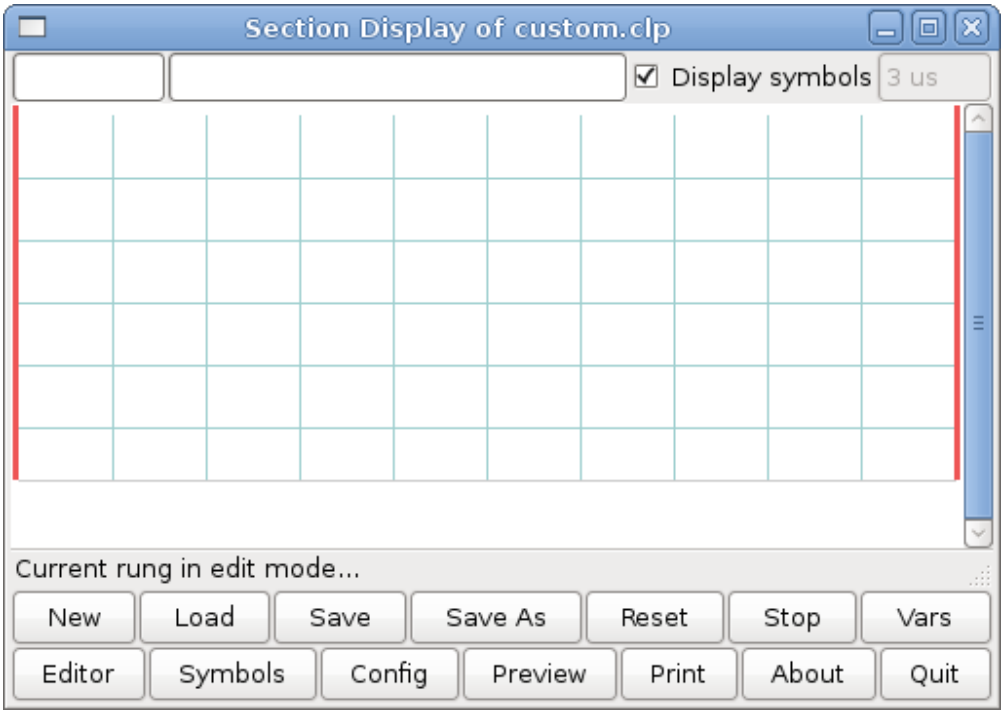


Figure 126. Abschnitt Display mit Grid

Klicken Sie nun auf den N.O.-Eingang im Editorfenster.

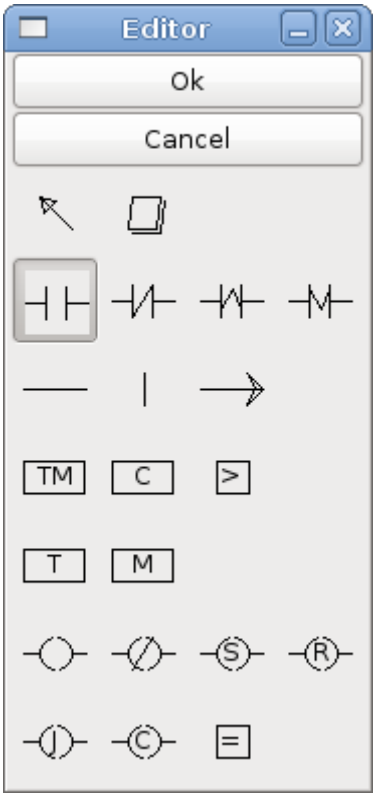


Figure 127. Editor-Fenster

Klicken Sie nun in das obere linke Raster, um den N.O.-Eingang in der Leiter zu platzieren.

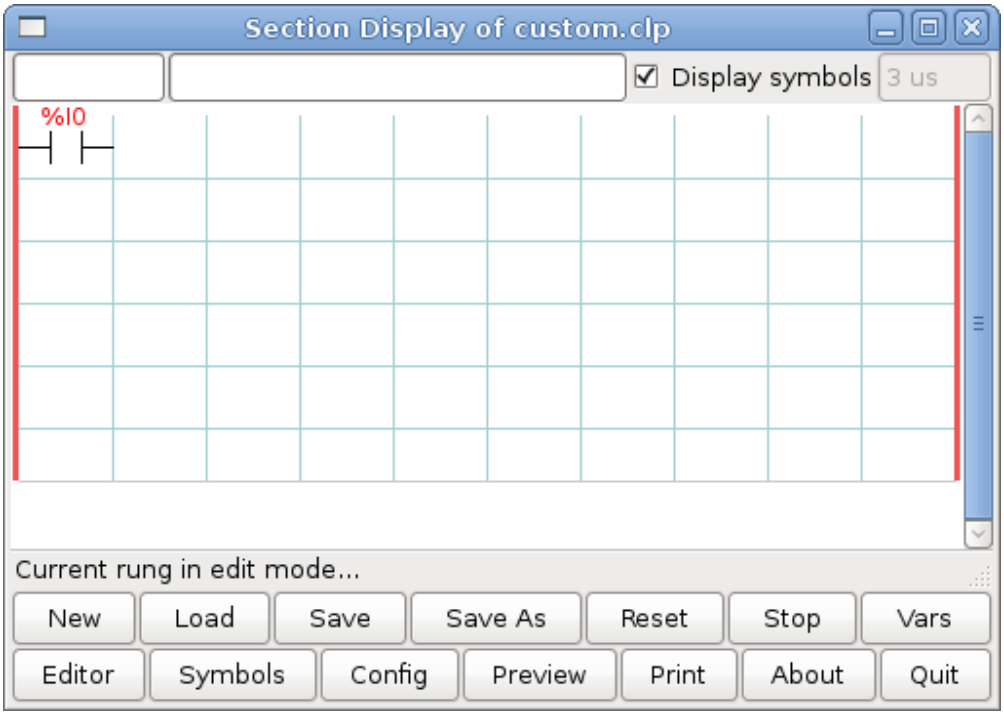


Figure 128. Abschnitt Display mit Input

Wiederholen Sie die obigen Schritte, um einen N.O.-Ausgang zum oberen rechten Gitter hinzuzufügen, und verwenden Sie die horizontale Verbindung, um die beiden zu verbinden. Es sollte wie folgt aussehen. Falls nicht, verwenden Sie den Radiergummi, um unerwünschte Abschnitte zu entfernen.

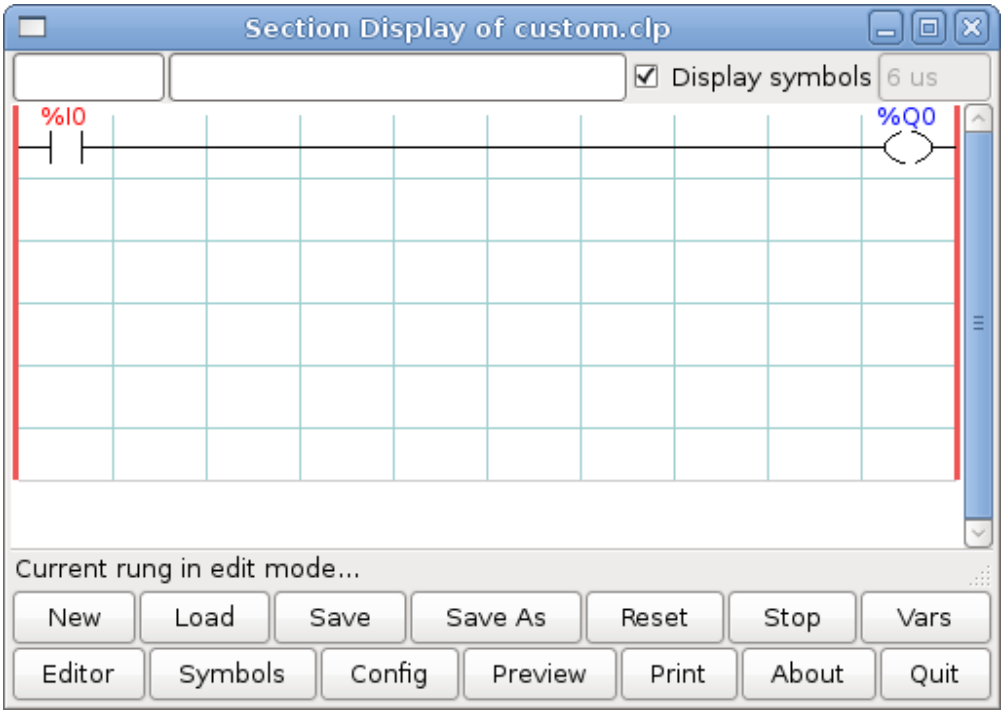


Figure 129. Abschnittsanzeige mit Sprosse

Klicken Sie nun im Editor-Fenster auf die Schaltfläche OK. Jetzt sollte Ihre Abschnittsanzeige wie folgt aussehen:

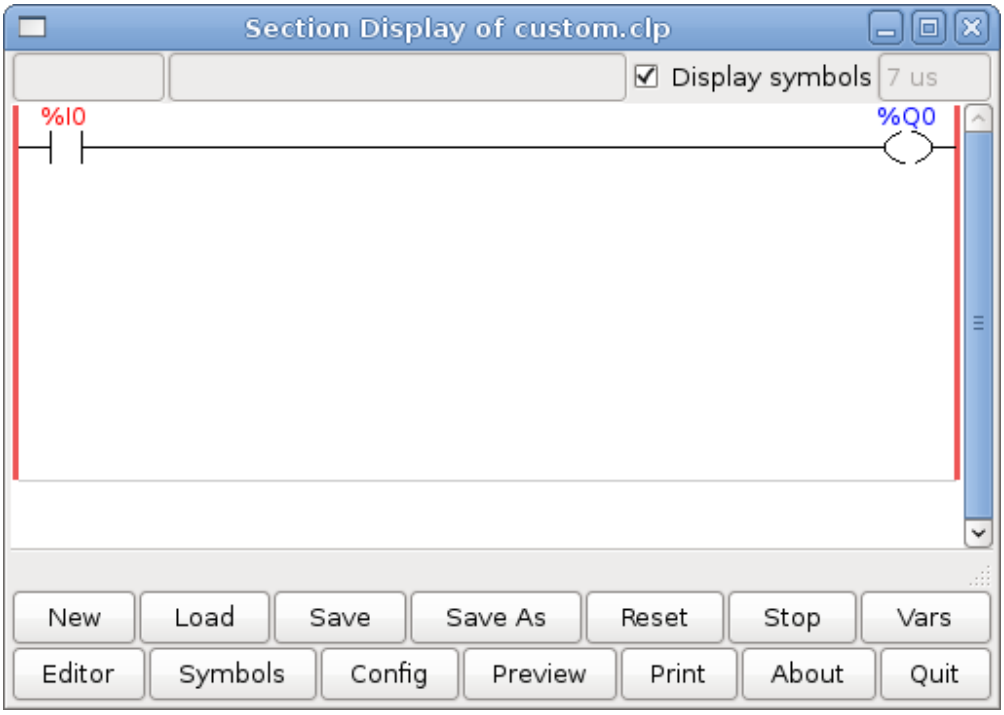


Figure 130. Abschnitt Anzeige Beendet

Um die neue Datei zu speichern, wählen Sie *Speichern unter* und geben Sie ihr einen Namen. Die Erweiterung .clp wird automatisch hinzugefügt. Als Speicherort sollte standardmäßig das laufende Konfigurationsverzeichnis angegeben werden.

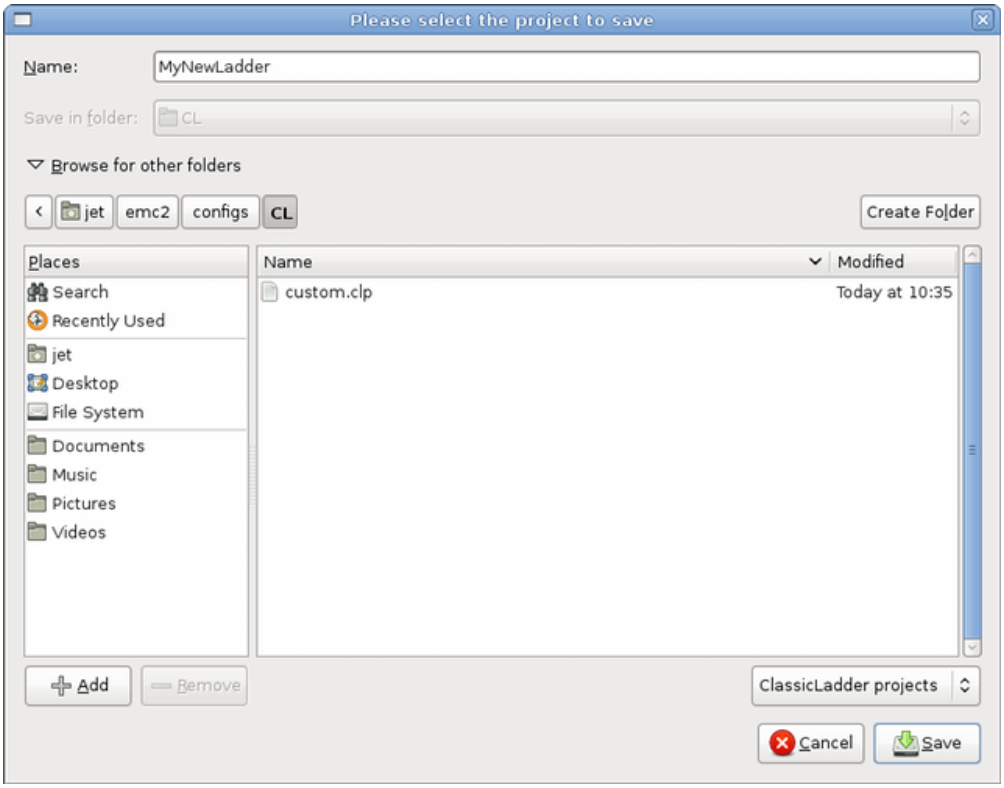


Figure 131. Speichern unter Dialog

Wenn Sie den StepConf-Assistenten zum Hinzufügen von ClassicLadder verwendet haben, können Sie auch diesen Schritt überspringen.

Um einen Leiter manuell hinzuzufügen, müssen Sie eine Zeile in Ihre custom.hal-Datei einfügen, die Ihre Leiterdatei lädt. Schließen Sie Ihre LinuxCNC-Sitzung und fügen Sie diese Zeile auf Ihre custom.hal Datei.

```
loadusr -w classicladder --nogui MyLadder.clp
```

Wenn Sie nun Ihre LinuxCNC-Konfiguration starten, wird auch Ihr Kontaktplanprogramm ausgeführt werden. Wenn Sie "Datei/Kontaktplan-Editor" wählen, wird das Programm, das Sie erstellt haben, im Fenster "Section Display" angezeigt.

8.3. ClassicLadder Beispiele

8.3.1. Umlaufender (engl. wrapping) Zähler

Um einen Zähler zu haben, der "umspringt", müssen Sie den Preset-Pin und den Reset-Pin verwenden. Wenn Sie den Zähler erstellen, setzen Sie den Preset auf die Zahl, die Sie erreichen wollen, bevor Sie auf 0 umbrechen. Die Logik ist, wenn der Zählerwert über dem Preset liegt, dann setzen Sie den Zähler zurück und wenn der Unterlauf an ist, dann setzen Sie den Zählerwert auf den Preset-Wert. Wie Sie im Beispiel sehen können, wird der Zähler zurückgesetzt, wenn der Zählerwert größer als der Vorwahlwert ist, und der Wert ist jetzt 0. Der Unterlaufausgang %Q2 setzt den Zählerwert auf den Vorwahlwert, wenn rückwärts gezählt wird.

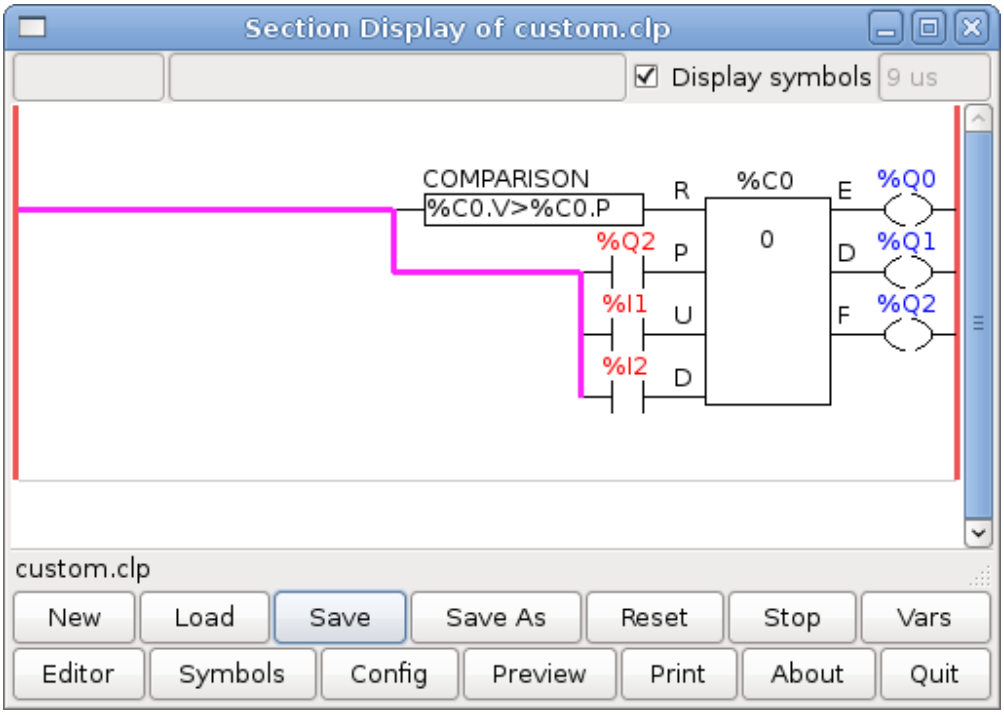


Figure 132. Umlaufender (engl. wrapping) Zähler

8.3.2. Extra-Impulse zurückweisen

Dieses Beispiel zeigt Ihnen, wie Sie zusätzliche Impulse von einem Eingang zurückweisen können. Nehmen wir an, der Eingangsimpuls %I0 hat die lästige Angewohnheit, einen zusätzlichen Impuls abzugeben, der unsere Logik stört. Der TOF (Timer Off Delay) verhindert, dass der zusätzliche Impuls unseren bereinigten Ausgang %Q0 erreicht. Das funktioniert so: Wenn der Timer einen Eingang erhält, ist der Ausgang des Timers für die Dauer der eingestellten Zeit eingeschaltet. Mit Hilfe eines Öffnerkontakts %TM0.Q blockiert der Ausgang der Zeitschaltuhr alle weiteren Eingänge, die unseren Ausgang erreichen, bis die Zeit abgelaufen ist.

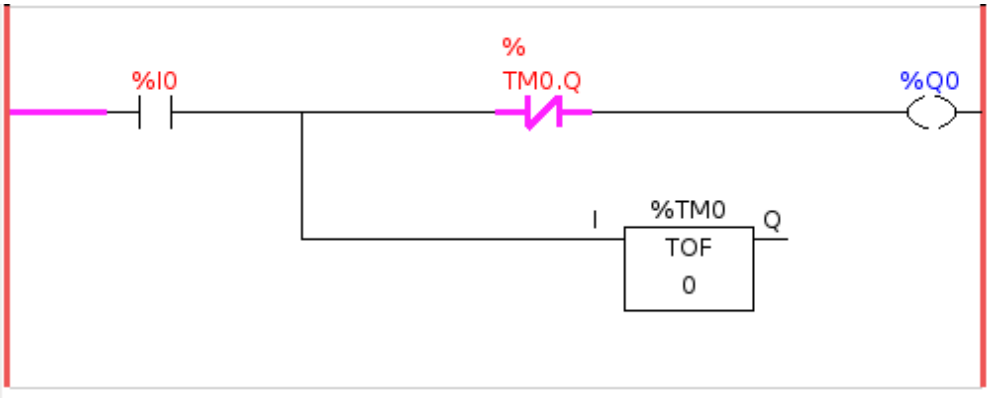


Figure 133. Extra-Impuls ablehnen

8.3.3. Externer Notaus

Das Beispiel für den externen Notaus-Schalter befindet sich im Ordner /config/classicladder/cl-estop. Es verwendet ein PyVCP-Panel, um die externen Komponenten zu simulieren.

Um eine externe Notaus-Schnittstelle zu LinuxCNC und haben die externen Notaus arbeiten zusammen

mit dem internen Notaus erfordert ein paar Verbindungen durch ClassicLadder.

Zuerst müssen wir die Notaus-Schleife in der Haupt-HAL-Datei öffnen, indem wir die folgenden Zeilen auskommentieren, indem wir das Doppelkreuz-Zeichen wie gezeigt hinzufügen oder sie entfernen.

```
# net estop-out <= iocontrol.0.user-enable-out
# net estop-out => iocontrol.0.emc-enable-in
```

Als Nächstes fügen wir ClassicLadder zu unserer Datei custom.hal hinzu, indem wir diese beiden Zeilen hinzufügen:

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

Als Nächstes führen wir unsere Konfiguration aus und erstellen die Leiter wie hier gezeigt.

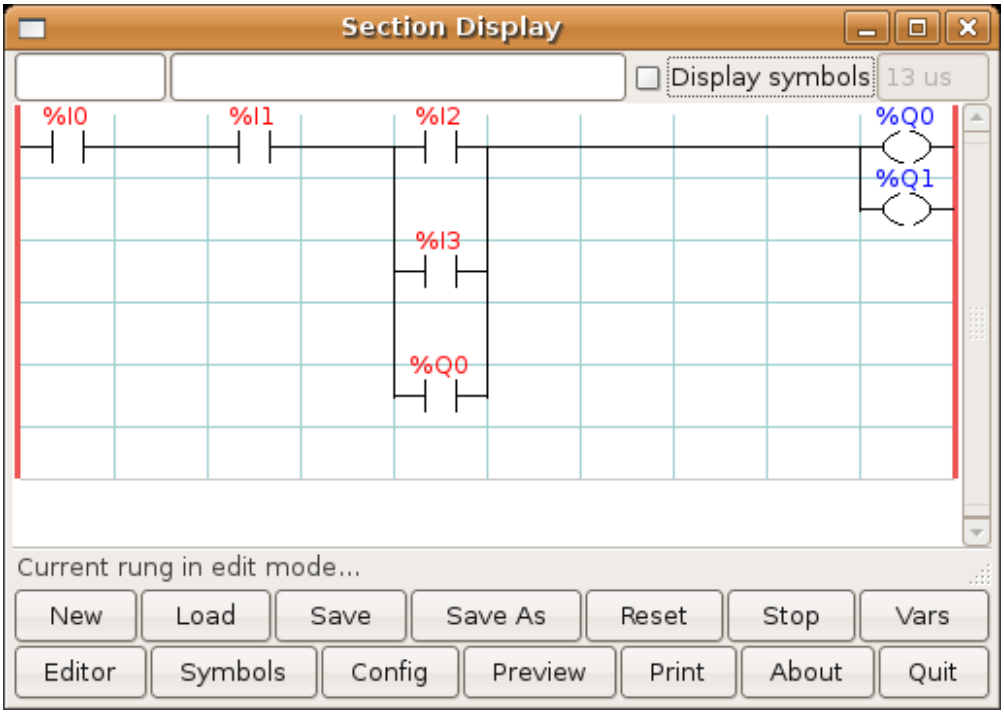


Figure 134. Anzeige des Notaus-Bereichs

Nach dem Erstellen der Leiter wählen Sie Speichern unter und speichern die Leiter als estop.clp

Fügen Sie nun die folgende Zeile in Ihre Datei custom.hal ein.

```
# Laden der Ladder
loadusr classicladder --nogui estop.clp
```

E/A-Zuweisungen

- %I0 = Eingabe aus dem PyVCP-Panel simulierten Notaus (die Checkbox)
- %I1 = Eingabe von LinuxCNC's Notaus
- %I2 = Eingang von LinuxCNC's Notaus Reset Impuls

- %I3 = Eingang von der PyVCP-Panel-Reset-Taste
- %Q0 = Ausgabe an LinuxCNC zur Freigabe
- %Q1 = Ausgang zum Freigabe-Pin der externen Treiberkarte (verwenden Sie einen N/C-Ausgang, wenn Ihre Karte einen Deaktivierungs-Pin hat)

Als nächstes fügen wir die folgenden Zeilen in die Datei custom_postgui.hal ein

```
# Beispiel für einen Notaus-Schalter mit PyVCP-Tasten zur Simulation externer Komponenten

# Der PyVCP-Checkbutton simuliert einen normalerweise geschlossenen externen Notaus-
Schalter.
net ext-estop classicladder.0.in-00 <= pyvcp.py-estop

# Anforderung der Notaus-Freigabe von LinuxCNC
net estop-all-ok iocontrol.0.emc-enable-in <= classicladder.0.out-00

# Anforderung der E-Stop-Freigabe von PyVCP oder einer externen Quelle
net ext-estop-reset classicladder.0.in-03 <= pyvcp.py-reset

# Diese Zeile setzt den Notaus von LinuxCNC zurück.
net emc-reset-estop iocontrol.0.user-request-enable => classicladder.0.in-02

# Diese Zeile ermöglicht es LinuxCNC, den Notausschalter in ClassicLadder zu entriegeln.
net emc-estop iocontrol.0.user-enable-out => classicladder.0.in-01

# Diese Zeile schaltet den grünen Indikator ein, wenn der E-Stop beendet ist.
net estop-all-ok => pyvcp.py-es-status
```

Als nächstes fügen wir die folgenden Zeilen in die Datei panel.xml ein. Beachten Sie, dass Sie die Datei mit einem Texteditor öffnen müssen, nicht mit dem Standard-HTML-Viewer.

```
<pyvcp>
<vbox>
<label><text>"Notaus Demo"</text></label>
<led>
<halpin>"py-es-status"</halpin>
<size>50</size>
<on_color>"green"</on_color>
<off_color>"red"</off_color>
</led>
<checkbutton>
<halpin>"py-estop"</halpin>
<text>"Notaus"</text>
</checkbutton>
</vbox>
<button>
<halpin>"py-reset"</halpin>
<text>"Reset"</text>
</button>
</pyvcp>
```

Starten Sie nun Ihre Konfiguration und sie sollte so aussehen.

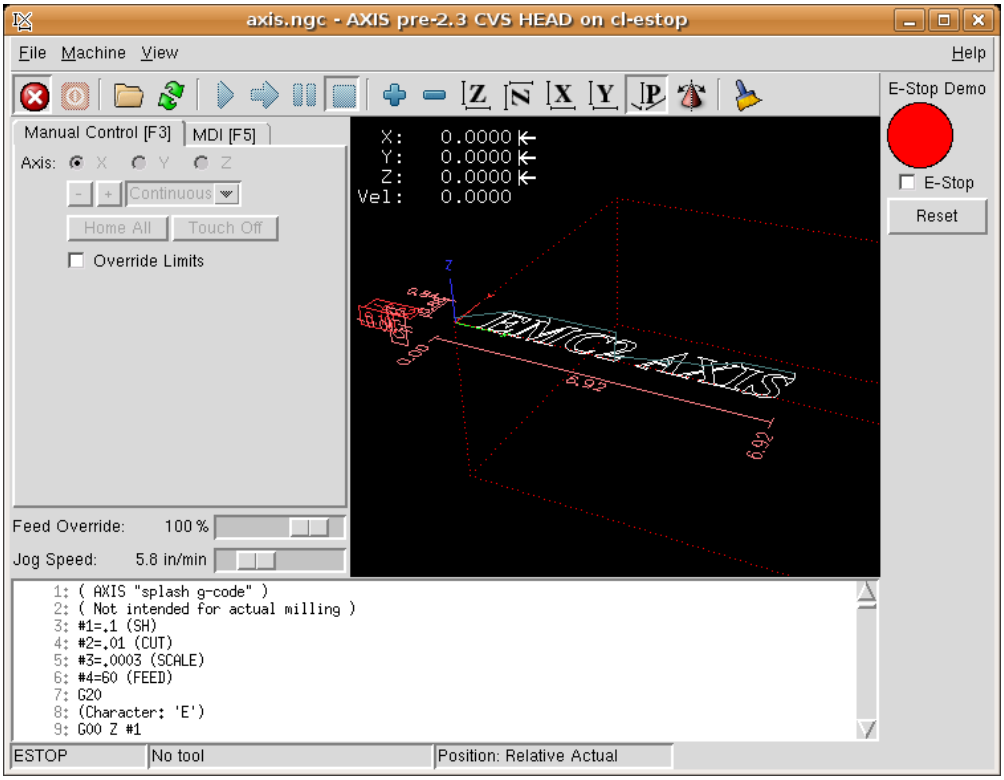


Figure 135. AXIS Notaus

Beachten Sie, dass Sie in diesem Beispiel wie im wirklichen Leben den ferngesteuerten Notaus (simuliert durch das Kontrollkästchen) deaktivieren müssen, bevor der AXIS Notaus oder der externe Reset Sie in den AUS-Modus versetzt. Wenn der Not-Aus-Schalter auf dem AXIS-Bildschirm gedrückt wurde, müssen Sie ihn erneut drücken, um ihn zu deaktivieren. Nach einem Notaus in AXIS können Sie keinen externen Reset durchführen.

8.3.4. Beispiel für Timer/Bedienung

In diesem Beispiel verwenden wir den Operate-Block, um der Timer-Voreinstellung einen Wert zuzuweisen, der davon abhängt, ob ein Eingang ein- oder ausgeschaltet ist.

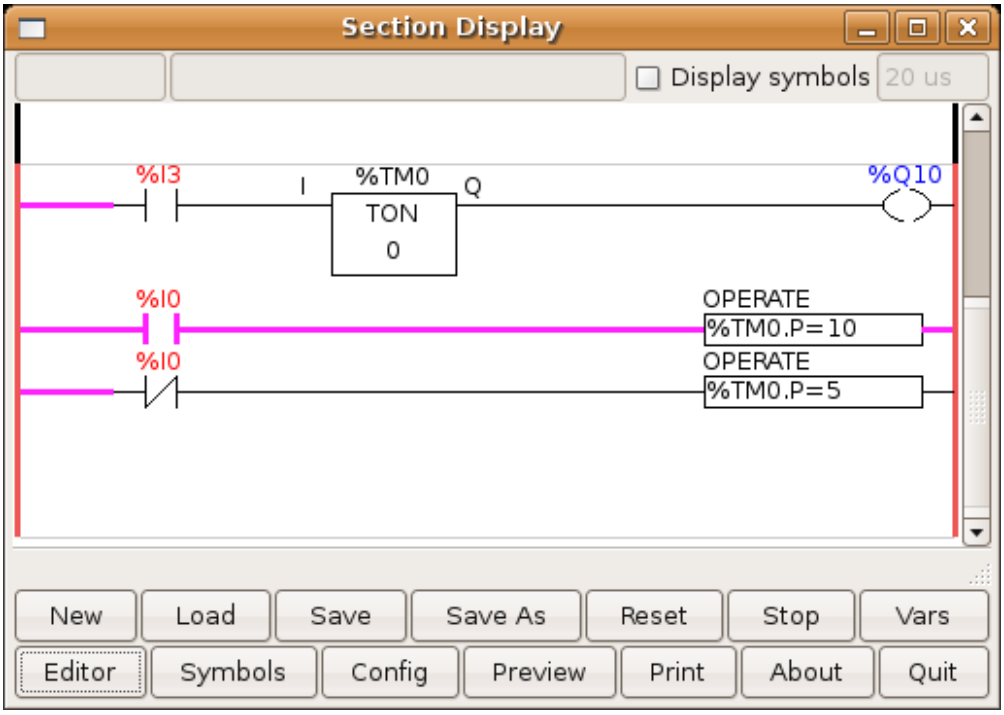


Figure 136. Beispiel für Timer/Bedienung

In diesem Fall ist %I0 wahr, so dass der voreingestellte Wert des Timers 10 ist. Wäre %I0 falsch, wäre der voreingestellte Zeitgeberwert 5.

Chapter 9. Fortgeschrittene Themen

9.1. Kinematik

9.1.1. Einführung

Wenn wir über CNC-Maschinen sprechen, denken wir in der Regel an Maschinen, die angewiesen werden, sich an bestimmte Orte zu bewegen und verschiedene Aufgaben auszuführen. Um eine einheitliche Sicht auf den Maschinenraum zu haben und ihn an die menschliche Sichtweise im 3D-Raum anzupassen, verwenden die meisten Maschinen (wenn nicht alle) ein gemeinsames Koordinatensystem, das kartesische Koordinatensystem.

Das kartesische Koordinatensystem besteht aus drei Achsen (X, Y, Z), die jeweils senkrecht zueinander stehen. footnote: [Das Wort "Achsen" wird auch häufig (und fälschlicherweise) verwendet, wenn von CNC-Maschinen die Rede ist, und bezieht sich auf die Bewegungsrichtungen der Maschine.].

Wenn wir über ein G-Code-Programm (RS274/NGC) sprechen, dann meist über eine Reihe von Befehlen (G0, G1 usw.), die Positionen als Parameter haben (X- Y- Z-). Diese Positionen beziehen sich genau auf kartesische Positionen. Ein Teil der LinuxCNC Motion Controller ist verantwortlich für die Übersetzung dieser Positionen in Maschinen-Positionen und deren Kinematik entsprechen [1].

Gelenke(engl. joints) vs. Achsen (engl. axes)

Ein Gelenk einer CNC-Maschine ist einer der physikalischen Freiheitsgrade der Maschine. Dies kann linear (Spindeln) oder rotierend (Drehtische, Roboterarmgelenke) sein. Es kann eine beliebige Anzahl von Gelenken an einer bestimmten Maschine geben. Ein beliebter Roboter hat beispielsweise 6 Gelenke, während eine typische einfache Fräsmaschine nur 3 hat.

Es gibt bestimmte Maschinen, bei denen die Gelenke so angeordnet sind, dass sie mit den kinematischen Achsen übereinstimmen (Gelenk 0 entlang der X-Achse, Gelenk 1 entlang der Y-Achse, Gelenk 2 entlang der Z-Achse); diese Maschinen nennt man Kartesische Maschinen (oder Maschinen mit Trivialkinematik). Diese Maschinen werden am häufigsten beim Fräsen verwendet, sind aber in anderen Bereichen der Maschinensteuerung (z. B. Schweißen: puma-typische Roboter) nicht sehr verbreitet.

LinuxCNC unterstützt Achsen mit Namen: X Y Z A B C U V W. Die X Y Z-Achsen beziehen sich normalerweise auf die üblichen kartesischen Koordinaten. Die A B C Achsen beziehen sich auf Rotationskoordinaten um die X Y Z Achsen. Die Achsen U V W beziehen sich auf zusätzliche Koordinaten, die üblicherweise kolinear zu den X-Y-Z-Achsen angeordnet sind.

9.1.2. Triviale Kinematik

Die einfachsten Maschinen sind solche, bei denen jedes Gelenk entlang einer der kartesischen Achsen angeordnet ist. Bei diesen Maschinen ist die Abbildung vom kartesischen Raum (das G-Code-Programm) auf den Gelenkraum (die tatsächlichen Aktoren der Maschine) trivial. Es handelt sich um eine einfache

1:1-Abbildung:

```
pos->tran.x = joints[0];
pos->tran.y = joints[1];
pos->tran.z = joints[2];
```

Im obigen Codeschnipsel kann man sehen, wie die Zuordnung erfolgt: die X-Position ist identisch mit dem Gelenk 0, die Y-Position mit dem Gelenk 1 usw. Die obige Darstellung bezieht sich auf die direkte Kinematik (eine Richtung der Transformation). Der nächste Codeschnipsel bezieht sich auf die inverse Kinematik (oder die umgekehrte Richtung der Transformation):

```
joints[0] = pos->tran.x;
joints[1] = pos->tran.y;
joints[2] = pos->tran.z;
```

In LinuxCNC wird die Identitätskinematik mit dem Kinematikmodul "trivkins" implementiert und auf 9 Achsen erweitert. Die Standardbeziehungen zwischen Achsenkoordinaten und Gelenknummern sind: [2] Fußnote:[Eine andere Möglichkeit, es zum Laufen zu bringen, besteht darin, den entsprechenden Code zu ändern und die Software neu zu kompilieren.]

```
pos->tran.x = joints[0];
pos->tran.y = joints[1];
pos->tran.z = joints[2];
pos->a      = joints[3];
pos->b      = joints[4];
pos->c      = joints[5];
pos->u      = joints[6];
pos->v      = joints[7];
pos->w      = joints[8];
```

Ähnlich sind die Standardbeziehungen für die inverse Kinematik für trivkins:

```
joints[0] = pos->tran.x;
joints[1] = pos->tran.y;
joints[2] = pos->tran.z;
joints[3] = pos->a;
joints[4] = pos->b;
joints[5] = pos->c;
joints[6] = pos->u;
joints[7] = pos->v;
joints[8] = pos->w;
```

Die Umwandlung für eine triviale "kins"-Kinematik oder eine kartesische Maschine ist einfach zu bewerkstelligen, sofern die verwendeten Achsenbuchstaben keine Lücken aufweisen.

Etwas komplizierter wird es, wenn der Maschine ein oder mehrere Achsenbuchstaben fehlen. Das Problem der fehlenden Achsenbuchstaben wird durch die Verwendung des Modulparameters *coordinates*= mit dem Modul trivkins gelöst. Jeder angegebenen Koordinate werden fortlaufend Gelenknummern zugewiesen. Eine Drehmaschine kann mit *coordinates=xz* beschrieben werden. Die Gelenkzuweisungen lauten dann:

```
joints[0] = pos->tran.x
joints[1] = pos->tran.z
```

Die Verwendung des Parameters *coordinates*= wird für Konfigurationen empfohlen, bei denen die Achsenbuchstaben weggelassen werden. Fußnote:[In der Vergangenheit unterstützte das Modul trivkins den Parameter *coordinates*= nicht, so dass Drehmaschinen-Konfigurationen oft als XYZ-Maschinen konfiguriert wurden. Die unbenutzte Y-Achse wurde so konfiguriert, dass sie 1) sofort in die Ausgangsposition fährt, 2) einen einfachen Loopback verwendet, um ihren Positionsbefehls-HAL-Pin mit ihrem Positionsrückmeldungs-HAL-Pin zu verbinden, und 3) in der Benutzeroberfläche nicht angezeigt wird. Zahlreiche Sim-Konfigurationen verwenden diese Methoden, um gemeinsame HAL-Dateien zu nutzen.]

Das Kinematikmodul *trivkins* erlaubt es auch, dieselbe Koordinate für mehr als ein Gelenk anzugeben. Diese Funktion kann bei Maschinen wie einem Portal mit zwei unabhängigen Motoren für die y-Koordinate nützlich sein. Eine solche Maschine könnte *coordinates=xyyz* verwenden, was zu Gelenkzuweisungen führt:

```
joints[0] = pos->tran.x
joints[1] = pos->tran.y
joints[2] = pos->tran.y
joints[3] = pos->tran.z
```

Weitere Informationen finden Sie auf den Manpages von trivkins.

9.1.3. Nicht-triviale Kinematik

Es gibt verschiedene Arten von Maschinenaufbauten (Roboter: Puma, Scara, Hexapods usw.). Jeder von ihnen ist mit linearen und rotierenden Gelenken ausgestattet. Diese Gelenke stimmen in der Regel nicht mit den kartesischen Koordinaten überein, daher benötigen wir eine Kinematikfunktion, welche die Umrechnung vornimmt (eigentlich 2 Funktionen: Vorwärts- und Rückwärtskinematikfunktion).

Zur Veranschaulichung der obigen Ausführungen werden wir eine einfache Kinematik namens Zweibein (eine vereinfachte Version des Dreibeins, das eine vereinfachte Version des Hexapods ist) analysieren.

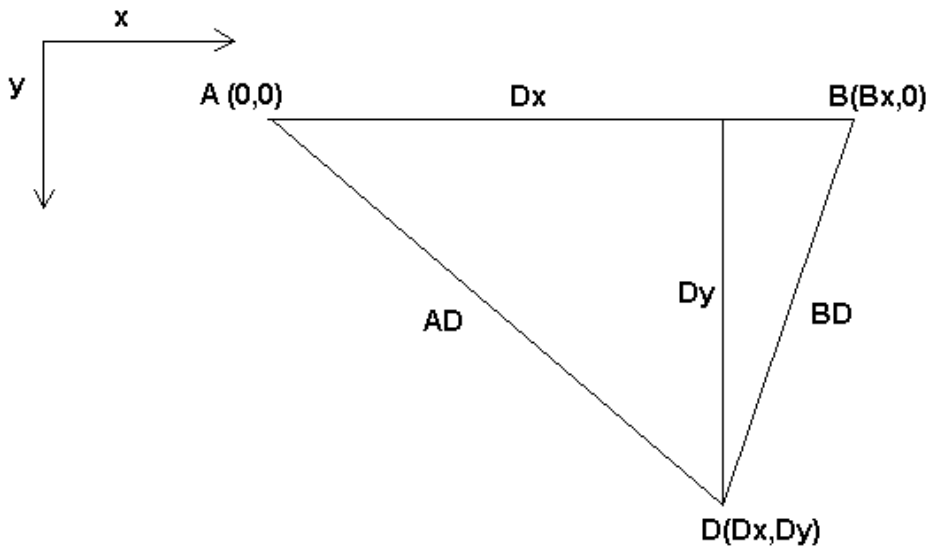


Figure 137. Zweibein-Einrichtung

Das Zweibein (engl. bipod), um das es hier geht, besteht aus zwei Motoren, die an einer Wand angebracht sind und an denen ein Gerät mit einem Draht aufgehängt ist. Die Gelenke sind in diesem Fall die Abstände zwischen den Motoren und dem Gerät (in der Abbildung mit AD und BD bezeichnet).

Die Position der Motoren ist per Konvention festgelegt. Motor A befindet sich in (0,0), was bedeutet, dass seine X-Koordinate 0 und seine Y-Koordinate ebenfalls 0 ist. Motor B befindet sich in (Bx, 0), was bedeutet, dass seine X-Koordinate Bx ist.

Unser Tooltip befindet sich im Punkt D, der durch die Abstände AD und BD und die kartesischen Koordinaten Dx, Dy definiert wird.

Die Aufgabe der Kinematik besteht darin, die Gelenklängen (AD, BD) in kartesische Koordinaten (Dx, Dy) und umgekehrt zu transformieren.

Vorwärts-Transformation

Um vom gemeinsamen Raum in den kartesischen Raum zu transformieren, werden wir einige trigonometrische Regeln anwenden (die rechtwinkligen Dreiecke, die durch die Punkte (0,0), (Dx,0), (Dx,Dy) und das Dreieck (Dx,0), (Bx,0) und (Dx,Dy) bestimmt werden).

Wir können leicht erkennen, dass:

$$AD^2 = x^2 + y^2 \; BD^2 = (Bx - x)^2 + y^2$$

ebenso:

$$BD^2 = (Bx - x)^2 + y^2$$

Wenn wir das eine von dem anderen abziehen, erhalten wir:

$$AD^2 - BD^2 = x^2 + y^2 - x^2 + 2 * x * Bx - Bx^2 - y^2$$

und deshalb:

$$x = \frac{AD^2 - BD^2 + Bx^2}{2 * Bx}$$

Daraus berechnen wir:

$$y = \sqrt{AD2 - x2}$$

Beachten Sie, dass die Berechnung von y die Quadratwurzel aus einer Differenz beinhaltet, was nicht unbedingt eine reelle Zahl ergibt. Wenn es keine einzige kartesische Koordinate für diese Gelenkposition gibt, dann wird die Position als Singularität bezeichnet. In diesem Fall liefert die Vorwärtskinematik den Wert -1.

Übersetzt in den tatsächlichen Code:

```
double AD2 = joints[0] * joints[0];
double BD2 = joints[1] * joints[1];
double x = (AD2 - BD2 + Bx * Bx) / (2 * Bx);
double y2 = AD2 - x * x;
if(y2 < 0) return -1;
pos->tran.x = x;
pos->tran.y = sqrt(y2);
return 0;
```

Inverse Transformation

Die inverse Kinematik ist in unserem Beispiel viel einfacher, da wir sie direkt schreiben können:

$$AD = \sqrt{x^2 + y^2}$$

$$BD = \sqrt{(Bx - x)^2 + y^2}$$

oder in tatsächlichen Code übersetzt:

```
double x2 = pos->tran.x * pos->tran.x;
double y2 = pos->tran.y * pos->tran.y;
joints[0] = sqrt(x2 + y2);
joints[1] = sqrt((Bx - pos->tran.x)*(Bx - pos->tran.x) + y2);
return 0;
```

9.1.4. Details zur Implementierung

Ein Kinematikmodul ist als HAL-Komponente implementiert und darf Pins und Parameter exportieren. Es besteht aus mehreren "C"-Funktionen (im Gegensatz zu HAL-Funktionen):

```
int kinematicsForward(const double *joint, EmcPose *world,
const KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

Implementiert die forward kinematics function.

```
int kinematicsInverse(const EmcPose * world, double *joints,
const KINEMATICS_INVERSE_FLAGS *iflags,
KINEMATICS_FORWARD_FLAGS *fflags)
```

Implementiert die Funktion der inversen Kinematik.

```
KINEMATICS_TYPE kinematicsType(void)
```

Gibt die Kennung des Kinematik-Typs zurück, typischerweise *KINEMATICS_BOTH*:

1. KINEMATICS_IDENTITY (jede Gelenknummer entspricht einem Achsenbuchstaben)
2. KINEMATICS_BOTH (Vorwärts- und Rückwärtskinematikfunktionen werden bereitgestellt)
3. KINEMATIKEN_FORWARD_ONLY
4. KINEMATICS_INVERSE_ONLY

NOTE GUIs können KINEMATICS_IDENTITY so interpretieren, dass die Unterscheidung zwischen Gelenknummern und Achsenbuchstaben im Gelenkmodus (typischerweise vor der Referenzfahrt) ausgeblendet wird.

```
int kinematicsSwitchable(void)
int kinematicsSwitch(int switchkins_type)
KINS_NOT_SWITCHABLE
```

Die Funktion kinematicsSwitchable() gibt 1 zurück, wenn mehrere Kinematiktypen unterstützt werden. Die Funktion kinematicsSwitch() wählt den Kinematik-Typ aus. Siehe Switchable Kinematitcs.

NOTE Die meisten Kinematikmodule unterstützen einen einzigen Kinematiktyp und verwenden die Direktive "**KINS_NOT_SWITCHABLE**", um Standardwerte für die erforderlichen Funktionen kinematicsSwitchable() und kinematicsSwitch() zu liefern.

```
int kinematicsHome(EmcPose *world, double *joint,
KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

Die Funktion home kinematics setzt alle ihre Argumente auf ihre richtigen Werte an der bekannten Ausgangsposition. Beim Aufruf sollten diese, sofern bekannt, auf Anfangswerte, z.B. aus einer INI-Datei, gesetzt werden. Wenn die Home-Kinematik beliebige Startpunkte akzeptieren kann, sollten diese Anfangswerte verwendet werden.

```
int rtapi_app_main(void)
void rtapi_app_exit(void)
```

Dies sind die Standardfunktionen zum Auf- und Abbauen von RTAPI-Modulen.

Wenn sie in einer einzigen Quelldatei enthalten sind, können Kinematikmodule mit *halcompile* kompiliert und installiert werden. Weitere Informationen finden Sie in der Manpage *halcompile(1)* oder

im HAL-Handbuch.

Kinematikmodul unter Verwendung der Vorlage userkins.comp

Eine weitere Möglichkeit, ein benutzerdefiniertes Kinematikmodul zu erstellen, ist die Anpassung der HAL Komponente *userkins*. Diese Vorlagenkomponente kann von einem Benutzer lokal geändert und mit halcompile erstellt werden.

Weitere Informationen finden Sie in den Man Pages von userkins.

Beachten Sie, dass zur Erstellung von schaltbaren kinematischen Modulen die erforderlichen Änderungen etwas komplizierter sind.

Siehe *millturn.comp* als Beispiel für ein umschaltbares kinematisches Modul, das mit der Vorlage *userkins.comp* erstellt wurde.

9.2. Kinematics Conventions

9.2.1. Einführung

A kinematics module answers two questions: where the controlled point is for a given set of joint positions, and which joint positions put it at a requested place. `kinematicsForward()` and `kinematicsInverse()` are those two answers.

Everything else about a machine's geometry is currently rebuilt outside the module. A tilted work plane needs the direction the tool points in, a simulation model needs the whole chain of frames, a limit check needs the rate joints move per unit of commanded motion. For the `xyzacb-trsrn` machine that geometry is written three times: as closed-form expressions in `src/hal/components/xyzacb_trsrn.comp`, as homogeneous matrices in the config's `remap_funcs_twp.py`, and as a chain of `HalRotate` calls with hand-chosen signs in `vismach/xyzacb-trsrn-gui.py`. The three copies share no vocabulary and no sign convention, so "the same as" can be checked only by a person reading all three.

This chapter fixes the vocabulary and states the rules, so that two modules written from the same drawing give the same answers and a consumer can say what it needs without naming a machine. Writing a Module collects the rules in one place.

9.2.2. Rahmen

Four frames, all right-handed.

Joint space

One coordinate per joint, in that joint's own units. This is what the motion controller commands and what `kinematicsForward()` is given.

Machine frame

Fixed to the machine bed, with the X, Y and Z of CNC Machine Overview. Nothing rotates it.

Work frame

Fixed to the workpiece. `kinematicsForward()` reports in it, and `kinematicsInverse()` is given it. Where no rotary carries the work it coincides with the machine frame. Where one does, it turns with the table and the module undoes that rotation, so that a G-code position keeps naming the same feature of the part however the table is set.

Tool frame

Fixed to the tool. Its origin is the controlled point of Controlled Point, the tool tip when a tool length offset is in effect and the tip of the spindle otherwise. Its third axis is the tool axis, The Tool Frame.

The interface calls the work frame "world": the forward and inverse take a `struct EmcPose *world`, and the motion controller speaks of world mode. That name is historical and stays in the code. This chapter says work frame, because the frame is attached to the workpiece and not to the room.

A pose is always a pair of frames, the moving one and the one it is measured against, and the two halves of this chapter use different pairs deliberately.

Positions are measured in the work frame, which is what makes a program independent of how the table is set.

Orientations are measured against the machine frame, and there are two of them. A module reports the tool frame and the work frame separately, each in machine coordinates. A consumer that wants the tool in workpiece coordinates composes them:

```
tool_in_work = transpose(work) * tool
```

A frame written as a matrix is the rotation whose columns are its axes in the coordinates it is measured against, so the tool axis in workpiece coordinates, the vector a tilted work plane asks the machine to reach and the one existing TWP code reads out of `matrix[0,2]`, `matrix[1,2]` and `matrix[2,2]`, is the third column of that product.

The pair is reported rather than the product because the product cannot be taken apart again, and a consumer that has to place both bodies needs each one against something that does not move. A simulation model draws the workpiece in one place and the tool in another; given only the product it can recover neither. Where only the tool turns, the work frame is the identity; where only the work turns, the tool frame is. A machine that turns both returns a real pair.

9.2.3. Rotation Sense

LinuxCNC states its rotation convention in Rotational Axes:

> The rotational axes are measured in degrees as wrapped linear axes in which the direction of positive rotation is counterclockwise when viewed from the positive end of the corresponding X, Y, or Z-axis. [...] Clockwise or counterclockwise is from the point of view of the workpiece.

The second sentence is the one that matters here. The rotation described is that of the tool relative to the

workpiece. Where the rotary carries the tool that is also the direction the physical axis turns; where it carries the work the table turns the other way, and the module converts between the two. This is the convention of ISO 841, which describes all motion as motion of the tool relative to the workpiece and primes the axes of a machine that moves the work: a table turning about Z is `+C'` when it produces the tool motion called `+C`.

So: given a pose whose C value increases, the tool moves counterclockwise about work Z seen from the workpiece, whichever member physically turns.

conventional-directions

`trtfuncs.c` and `maxkins.c` carry a `conventional-directions` HAL pin that selects the sign of the rotary terms, and default it to false, which is the opposite sense. Existing configurations keep working; new configurations set it true.

Leaving it false costs two things. The direction a program runs in depends on a HAL pin rather than on the G-code. And the rotary values the module reports are the raw joint values,

```
pos->a = joints[JA];
pos->c = joints[JC];
```

while the translations in the same call were computed with the opposite sign, so the returned pose does not describe its own orientation. A caller cannot rebuild the tool frame from `pos->a`, `pos->b` and `pos->c` without separately knowing how the pin is set. That is the immediate reason the tool frame has to be an answer from the module rather than something a caller derives from the pose.

9.2.4. The Tool Frame

The tool axis

The tool axis is the third axis of the tool frame. It points from the tip towards the holder, away from the material.

It is a direction, not a distance, and is unrelated to the tool length: the length is the scalar the `tool-length` pin carries, and the tool axis is the direction that length is applied along.

Where it points for a given joint set is whatever the machine's geometry makes it. A plain vertical mill has `[0, 0, 1]` in machine coordinates at all times, and a machine whose spindle is parallel to Z with its rotaries at zero has it there too, but that is a property of those machines and not a rule. `pumakins` with its supplied parameters has `[0, 0, -1]` with every joint at zero, and `genserkins` takes its Denavit-Hartenberg parameters from HAL pins, so for that module the question has no fixed answer at all.

Native frames that point the other way

Robot kinematics name the same line in the opposite sense. ISO 9787 clause 5.3 places the mechanical interface coordinate system at the centre of the flange, where the "+Zm axis points perpendicularly away from the mechanical interface", which runs holder towards tip. The Denavit-Hartenberg approach vector is the same sense. `pumakins` builds that frame and uses it, reaching the tip by adding the tool

length along the third column:

```
hom.tran.x = hom.tran.x + hom.rot.z.x*PUMA_D6;
```

Both senses come from a standard for a class of machine, and the tree contains both. What a module reports is the machine tool sense, tip towards holder, because that is the direction a tilted work plane commands and what machine Z already means to a mill operator.

Turning one sense into the other is not a change of sign. Negating the third column leaves a matrix of determinant -1, a reflection, which is not a frame any machine can hold. Reversing the tool axis and staying right-handed takes a half turn about one of the two transverse axes, and which one is chosen decides where tool X lands.

Because it is a rotation in its own right, a module declares it rather than applying it by hand, as the last argument of `switchkinsRegisterFrames()`. Shared code applies it and checks once, at load, that it is orthonormal with determinant +1. `TOOL_FRAME_SPINDLE` is the identity, for a module whose maths is already in the convention; `TOOL_FRAME_FLANGE` is the half turn a Denavit-Hartenberg module needs. Keeping it in one place makes it greppable and stops the next such module quietly choosing the other half turn, which would be right about the tool axis and wrong about tool X.

The declaration is fixed when the module is written and nothing changes it at runtime. A machine whose native frame moved while running would be a machine whose geometry moves underneath the program.

Tool X

A tilted work plane commands only where the tool points. Reaching that direction uses both rotaries of a five-axis machine, and the rotation of the tool about its own axis is then whatever the chain leaves rather than anything the program chose. For cutting that does not matter, the cutter being a solid of revolution. It matters as soon as the tool frame is used as a coordinate system for programming, which is what `G68.2` does: the operator writes X and Y moves in the tilted plane and has to know where its X points.

So the software places it, through a virtual rotation about the tool axis applied after the physical joints. It is the `pre-rot` pin on the in-tree kinematics components and `virtual_rot` in the TWP code, one quantity under two names. A machine with no such pin has no say in the matter: its tool X is whatever the chain produces, and a consumer that needs a defined one applies the rotation itself.

The frame a module reports is the joints' alone and does not carry the virtual rotation. The rotation is a rotation of the coordinate system, not of the machine, and that is where the other controls keep it: Heidenhain's `COORD ROT` rotates the working plane and Fanuc's `G68.2` feature coordinate system carries its own X. Reporting the joints alone also keeps the orientation inverse honest: the turn it reports about the tool axis is then the value to apply, not a difference from whatever is applied already. A consumer that wants tool X as programmed, a preview or a model, multiplies the frame by the rotation on the pin.

IMPORTANT	By default, tool X lies parallel to the machine XY plane. Where the tool axis is vertical and that leaves tool X free, tool X is machine X. `G68.3 R` rotates the

> frame from there.

This fixes tool X only up to a half turn, two opposite directions both being horizontal. Where a module has to choose, it takes the one that keeps the frame continuous with the previous pose.

Deriving the default rotation

The convention is stated rather than a formula because the formula differs from machine to machine and follows from the convention. Write the tool orientation as the product of the primary, secondary and virtual rotations:

```
M = Rp(theta_1) * Rs(theta_2) * Rz(tc)
```

Tool X is the first column of `M`, so "tool X is horizontal" is the statement that `M[2][0]` is zero. Solving that for `tc` gives the default.

For a nutating head of nutation angle `v`, writing `Sv = sin(v)`, `Cv = cos(v)`, `Ss = sin(theta_2)`, `Cs = cos(theta_2)`, `s = Cs + Cv*Cv*(1 - Cs)` and `t = Sv*Cv*(1 - Cs)`, the two nutating machines in the tree have different secondary rotations, and so different bottom rows:

machine	bottom row of `Rs`	resulting default
`xyzacb-trsrn`	`[-Sv*Ss, t, s]`	`tc = atan2(Sv*Ss, t)`
`xyzbca-trsrn`	`[t, Sv*Ss, s]`	`tc = atan2(-t, Sv*Ss)`

The two formulas look unrelated and are the same rule. Apply either to the other machine and the result is a frame whose tool **Y** is horizontal, a quarter turn from what was wanted, and no test in the tree notices: the tool still points where it was told to point, and only the meaning of X and Y in the tilted plane has changed. A module that documents its `Rs` and cites this rule can be checked. One that documents only its `tc` formula cannot.

9.2.5. What a Module Reports

Position, through `kinematicsForward()` and `kinematicsInverse()`, in the work frame.

Orientation, through `kinematicsWorkFrame()` and `kinematicsToolFrame()`, each against the machine frame, as Frames describes. The consumers are tilted work plane handling, tool length compensation along a tilted axis, previews and simulation models, and probing routines that have to say which way the stylus faces.

The joint values that reach a requested orientation, through `kinematicsToolFrameInverse()`, which is the inverse of the pair above and the question a tilted work plane asks when it has to orient the machine. The Orientation Inverse says what it answers.

How joint motion follows world motion, through `kinematicsJacobian()`, so that a feed can be checked against the joint velocity and acceleration it will actually demand, and so that proximity to a singularity is a number rather than a surprise. The Jacobian says what it answers and in which units.

All of these are functions of the joint values and the module's own geometry. None needs state carried between calls, and none needs the module to be running in a realtime thread to be useful: the interesting callers, a limit check before a move and a preview before a program runs, are not in the servo loop.

9.2.6. The Orientation Inverse

Pointing the tool somewhere is the question a tilted work plane asks on every orienting move, and the question a program written as tool vectors asks on every block. It is the inverse of the tool frame: not where the tool points for these joints, but which joints point it there.

It has more than one answer. A five-axis machine reaches a given tool axis two ways, one with the secondary rotary positive and one with it negative, and which of the two is wanted depends on the joint limits and on what the operator asked for, shortest move or one direction of rotation only. So the module reports every set of joint values that reaches the orientation and stops there. It does not apply the joint limits and it does not rank the answers, because neither is geometry: a module that picked for the caller would be picking with less to go on than the caller has.

The request is a tool axis, and optionally a tool x as well. The two have to be at right angles, being two axes of one frame.

Asking for tool x does not require a joint that can reach it. A five-axis machine spends both rotaries on the tool axis, and the turn about that axis is not a joint at all: it is the virtual rotation of The Tool Frame. So the answer comes back in two parts. Where the joints can place tool x, on a machine with a third orientation joint, they do, and the reported turn is zero. Where they cannot, the joints reach the axis and the reported turn finishes the job. The caller writes one path either way, and which kind of machine it has is a number that happens to be zero rather than a branch.

This is what the controls do. A Heidenhain `PLANE VECTOR` block carries the normal and the base vector together, the normal reached by the rotaries and the base vector applied as a rotation of the coordinate system; `SEQ` then picks between the rotary solutions. A Fanuc `G68.2` defines the plane including where its X points, and `G53.1` moves the rotaries to align the tool axis. Neither refuses a program for naming both directions on a five-axis machine, and neither should this.

Some requests still do not pin the machine down. A five-axis machine asked to point the tool along the axis its primary rotary turns about can hold any primary angle; a machine with three orientation joints asked only for a tool axis has a whole curve of solutions. In both cases the answer is a continuum, so one point of it is reported, the one nearest where the machine already is, along with the number of directions left free. A caller that hands back a list of samples from a curve as though they were alternatives is telling the operator something false.

Which joints may take part is the caller's decision, not the module's. A rotary table turns the tool against the work as surely as a head rotary does, so a machine with a table and a two-axis head has three orientation joints and a bare tool axis leaves one of them free. Whether the table is used for that is a machining choice: Heidenhain names the axes a tilt may use with `M138`, and its `PLANE` lets the operator pick `TABLE ROT`, turn the table to place tool X, or `COORD ROT`, leave the table and rotate the coordinate system. So the request names the joints to hold. Holding the table gives the two head solutions and the turn about the tool that finishes the frame, which is what the TWP remap computes; holding nothing lets

the table place tool X and the turn comes back zero.

The request is taken as a program carries it. A direction written to a few digits is a unit vector only to within its rounding, and a tool X written the same way is at right angles to the axis only to within it, so both are normalised on the way in and only a vector that is not a direction at all is refused.

What a module has to supply

Nothing, if it already reports its frames. The shared code answers the question by searching: it finds which joints move `transpose(work) * tool`, and solves for them. That is the whole reason the frames are worth reporting. A module that supplies them gets the inverse without deriving anything.

A module with a closed form registers it and that is used instead. It is faster than a search and it knows its own degenerate poses without having to discover them. The nutating heads are the case in point: for those,

```
cos(secondary) = (Kzz - Cv^2) / (1 - Cv^2)
```

with `Cv = cos(v)` for a nutation angle `v` and `Kzz` the z component of the requested tool axis, and the primary follows from the other two components. Both roots of the arc cosine are solutions, which is where the pair of answers comes from.

The search is not a realtime routine. How long it takes depends on the machine and on the request, and the callers that want it, orienting a tilted work plane and previewing a program, are not in the servo loop.

9.2.7. The Jacobian

A feed is a speed in the work frame. What the machine has to deliver is a speed at each joint, and on any kinematics that is not the identity the two are related by where the machine is. The Jacobian is that relation at one pose: how each joint responds to a unit rate of each pose coordinate.

```
jac[j][a] = d joint[j] / d pose[a]
```

Rows are joints. Columns are the pose coordinates in `EmcPose` order, X Y Z A B C U V W. It is the derivative of `kinematicsInverse()`: multiplied by a pose velocity it gives the joint velocity motion will command, which is what a feed limit compares with the joint limits. Joint `j` binds when

```
|jac[j] . tangent| * F
```

exceeds that joint's velocity limit, `tangent` being the direction of the move in pose coordinates and `F` the feed along it. The acceleration limit follows from a second Jacobian taken further along the path, with no more from the module. A row that grows without bound is a pose approaching a singularity, where no world speed is slow enough for the joints to follow.

Einheiten

Each entry is in joint units per pose unit, whatever units the module's own forward and inverse already use. Nothing is converted: a caller that feeds pose rates in `EmcPose` units gets joint rates in the units motion already commands, and never has to know which unit a rotary joint is in. On every module in the tree both are degrees, so a table rotary's own row is a 1 in its own column, and a robot's rotary rows carry degrees per millimetre against the linear columns.

This is why the Jacobian, unlike the orientation inverse, does not need the interface to name the rotary joint unit. Every number in it is a ratio of quantities that already pass through `kinematicsForward()` and `kinematicsInverse()`, and the caller never combines it with anything measured in another unit.

Frame

The columns are pose coordinates, so the answer lives in the work frame, where `kinematicsForward()` reports positions. The A, B and C columns are rates of the pose words, the wrapped linear axes the planner already treats as coordinates, and not an angular velocity vector: on a machine that carries the work the forward writes the rotary joint into the pose word, and that column says exactly that, a 1 for its own joint.

That makes this a different object from the frames of Frames, and the two rules are kept apart deliberately. A frame is an orientation, and a renderer placing two bodies needs each against something fixed, so frames are reported against the machine. A Jacobian is a derivative of the pose, and everything that uses it multiplies it by a pose rate, so it is reported where the pose is. A module whose maths produces a twist in the machine frame, which is what the Denavit-Hartenberg modules produce, turns it into pose word rates through the matrix of the axes each pose word turns about, once, inside the module. `genserkins` does this, and having it written once there is worth more than the closed form itself, since every consumer would otherwise guess it.

What a module has to supply

Nothing. If a module registers no Jacobian, the shared code computes one by central differences: it calls the module's inverse eighteen times, stepping each pose coordinate a small amount to either side on the solution branch the inverse flags select, and differences the results. The answer is as precise as the inverse itself; behind an inverse that iterates, that is the iteration's convergence tolerance divided by the step. Modules built on `switchkins.c` answer this way for every type that registers nothing; an identity type answers exactly.

This cost matters because `kinematicsJacobian()` is meant to run in the servo thread, where checking a feed limit calls it once per cycle: each call then costs eighteen inverses. On the closed-form inverses in the tree that is under two microseconds. `genserkins`, the one module whose inverse iterates, would cost near eighty microseconds per call this way, against under two for the geometric Jacobian it registers instead.

So a module whose inverse iterates should register a closed-form Jacobian, as below. Nothing enforces this. A module that registers nothing still gets a correct Jacobian, just one that costs eighteen inverses and carries the iteration's precision, which behind an iterating inverse is too slow for the servo thread.

A module with a closed form registers it with `switchkinsRegisterJacobian()`. It is exact, it costs

what the inverse costs, and it knows its own singular poses rather than discovering them as an inverse that fails a step away from the pose. Every module in the tree supplies one. The two arms whose inverse is a chain of arc tangents, `pumakins` and `three21kins`, take theirs from their Denavit-Hartenberg chain through `kinsJacobianFromDhArm()`, which any six-joint serial arm can call with its own table, written in the modified convention of Craig's Introduction to Robotics: each joint's axis crossed with the vector from it to the tool point gives the point's rate, the axis itself the angular rate, and the six by six that makes is inverted and taken through the roll, pitch and yaw rates the pose words are.

A module reading its rotary angles from the joint argument of the inverse rather than from the pose, which the nutating heads do, has an inverse whose derivative about the pose is not the coupling the machine has. Such a module supplies the closed form, taken against the pose.

9.2.8. Writing a Module

Rahmen

Report positions in the work frame, so a rotary that carries the work is undone in `kinematicsForward()`. Report the work frame and the tool frame separately, each against the machine frame, so that a consumer placing both bodies can.

Signs

Positive A, B and C are counterclockwise about work X, Y and Z viewed from the positive end, describing the motion of the tool relative to the workpiece. A new module does not offer a pin that reverses this.

Tool axis

Tip towards holder. A module whose maths is written with the Denavit-Hartenberg approach vector declares the half turn that relates the two, rather than applying it by hand.

Tool X

The default virtual rotation puts tool X parallel to the machine XY plane. Derive the value from the module's own rotation matrices, and write those matrices down in the module.

Orientation inverse

A module that reports its frames gets it from the shared search and needs to do nothing. Register a closed form only where one exists, and where it does, say which poses it treats as degenerate.

Jacobian

Rows are joints, columns are pose coordinates, entries in the units the forward and inverse already use, reported where the pose is. A module with a closed form inverse differentiates it and registers the result; one without lets the shared code difference the inverse.

Geometry stays in the module

Whatever a consumer needs to know about the machine's shape is answered by the module. A consumer that restates it has taken a copy that nothing keeps in step, which is the situation this chapter exists to end.

Mount orientation is not this

A tool or holder mount orientation is a different quantity: a right-angle head, a tool held at an angle, an end effector clocked on its flange. Those vary from setup to setup and belong with the rest of the tool data, addressed from the program where the interpreter can see them, and not in HAL alone where lookahead and preview cannot see them and where they can move underneath a running program. The tool table and `G43.1` already carry per-tool A, B and C words, which is the right addressing route, but they shift the rotary axis reading and configurations depend on that, so a mount orientation is a new field rather than a reinterpretation of that one. It wants to be stated as a frame, which can be checked for orthonormality and determinant, rather than as three angles whose ordering convention is written down nowhere.

9.2.9. Referenzen

- CNC Machine Overview, for the axis nomenclature and the rotation convention this chapter builds on.
- 5-Axis Kinematics, for worked transformations of the table-rotary and tilting-table configurations.
- Switchable Kinematics, for how a machine presents more than one of these models at once.
- ISO 841, Industrial automation systems and integration, Numerical control of machines, Coordinate system and motion nomenclature.
- ISO 9787, Robots and robotic devices, Coordinate systems and motion nomenclatures, clause 5.3, for the flange frame the robot modules follow.

9.3. Einrichten "modifizierter" Denavit-Hartenberg (DH)-Parameter für "genserkins"

9.3.1. Vorspiel

LinuxCNC unterstützt eine Reihe von Kinematik-Module, einschließlich einer, die eine verallgemeinerte Reihe von seriellen Kinematik allgemein über Denavit-Hartenberg Parameter angegeben unterstützt.

Dieses Dokument veranschaulicht eine Methode, um die DH-Parameter für eine Mitsubishi RV-6SDL in LinuxCNC mit *genserkins* Kinematik eingerichtet.

NOTE Dieses Dokument befasst sich nicht mit der Erstellung eines "Vismach"-Modells, das zwar sehr nützlich ist, aber eine ebenso sorgfältige Modellierung erfordert, wenn es dem in diesem Dokument abgeleiteten "Genserkins"-Modell entsprechen soll.

NOTE Es kann Fehler und/oder Mängel geben - Nutzung auf eigene Gefahr!

9.3.2. Allgemeines

Mit der zunehmenden Verbreitung von Industrierobotern steigt auch das Interesse, die verwendeten

Roboter mit LinuxCNC zu steuern. Eine häufige Art von Roboter in der Industrie und Fertigung verwendet wird, ist die "serielle Manipulator" als eine Reihe von motorisierten Gelenke durch starre Verbindungen verbunden konzipiert. Serienroboter haben oft sechs Gelenke, die für die sechs Freiheitsgrade erforderlich sind, um ein Objekt im Raum zu positionieren (XYZ) und zu orientieren (ABC oder Nick, Roll, Gier). Oft haben diese Roboter eine Armstruktur, die sich von einer Basis bis zu einem Endeffektor erstreckt.

Die Steuerung eines solchen Serienroboters erfordert die Berechnung der Position und Ausrichtung des Endeffektors in Bezug auf ein Referenzkoordinatensystem, wenn die Gelenkwinkel bekannt sind (**vorwärtsgerichtete Kinematik**), sowie die komplexere umgekehrte Berechnung der erforderlichen Gelenkwinkel für eine bestimmte Position und Ausrichtung des Endeffektors in Bezug auf das Referenzkoordinatensystem (**inverse Kinematik**). Die mathematischen Standardwerkzeuge, die für diese Berechnungen verwendet werden, sind Matrizen, d. h. Tabellen mit Parametern und Formeln, die den Umgang mit den Rotationen und Translationen erleichtern, die bei der Berechnung der Vorwärts- und Rückwärtskinematik erforderlich sind.

Detaillierte Kenntnisse der Mathematik sind für einen Serienroboter nicht erforderlich, da LinuxCNC ein Kinematikmodul bereitstellt, das einen Algorithmus namens "genserkins" implementiert, um die Vorwärts- und Rückwärtskinematik für einen generischen Serienroboter zu berechnen. Um einen bestimmten Serienroboter zu steuern, muss *genserkins* mit Daten versorgt werden, so dass es ein mathematisches Modell der mechanischen Struktur des Roboters aufbauen und damit die Mathematik tun kann.

Die erforderlichen Daten müssen in einer standardisierten Form vorliegen, die von Jacques Denavit und Richard Hartenberg bereits in den fünfziger Jahren eingeführt wurde und als DH-Parameter bezeichnet wird. Denavit und Hartenberg verwendeten vier Parameter, um zu beschreiben, wie ein Gelenk mit dem nächsten verbunden ist. Diese Parameter beschreiben im Wesentlichen zwei Rotationen (*alpha* und *theta*) und zwei Translationen (*a* und *d*).

9.3.3. Modifizierte DH-Parameter

Wie so oft wurde dieser "Standard" von anderen Autoren modifiziert, die "modifizierte DH-Parameter" eingeführt haben, und man muss sehr vorsichtig sein, denn "genserkins" verwendet "modifizierte DH-Parameter", wie sie in der Veröffentlichung "Introduction to Robotics, Mechanics and Control" von John J. Craig beschrieben sind. Vorsicht, es gibt viele Informationen zu "DH-Parametern", aber selten definiert der Autor, welche Konvention tatsächlich verwendet wird. Darüber hinaus haben einige Leute es für nötig befunden, den Parameter mit der Bezeichnung "a" in "r" zu ändern und damit zur Verwirrung beigetragen. Dieses Dokument hält sich an die Konvention in der oben erwähnten Veröffentlichung von Craig, mit dem Unterschied, dass die Aufzählung der Fugen und Parameter mit der Zahl *0* beginnt, um mit *genserkins* und seinen HAL-Pins konsistent zu sein.

Standard- und modifizierte DH-Parameter bestehen aus vier numerischen Werten für jedes Gelenk ("a", "d", "alpha" und "theta"), die beschreiben, wie das Koordinatensystem (CS), das in einem Gelenk sitzt, bewegt und gedreht werden muss, um mit dem nächsten Gelenk ausgerichtet zu werden. Ausgerichtet bedeutet, dass die Z-Achse unseres Koordinatensystems mit der Rotationsachse des Gelenks zusammenfällt und in die positive Richtung zeigt, so dass die Finger in die positive Drehrichtung des Gelenks zeigen, wenn man die Regel der rechten Hand anwendet und der Daumen in die positive Richtung der Z-Achse zeigt. Es wird deutlich, dass man dazu die positiven Richtungen aller Gelenke

festlegen muss, bevor man mit der Ableitung der Parameter beginnt!

Der Unterschied zwischen der "Standard"- und der "modifizierten" Notation besteht darin, wie die Parameter den Verbindungen zugewiesen werden. Die Verwendung der "Standard" DH-Parameter in "genserkins" ergibt **nicht** das korrekte mathematische Modell.

9.3.4. Modifizierte DH-Parameter, wie sie in *Genserkins* verwendet werden

Beachten Sie, dass *genserkins* keine Offsets auf Theta-Werte behandelt — Theta ist die Gelenkvariable, die von LinuxCNC **kontrolliert** wird. Mit dem CS mit dem Gelenk ausgerichtet, ist eine Drehung um seine Z-Achse identisch mit der Drehung befohlen, dass das Gelenk von LinuxCNC. Dies macht es unmöglich, die 0° Position unserer Roboter Gelenke willkürlich zu definieren.

Die drei konfigurierbaren Parameter sind:

1. **alpha** : positive oder negative Drehung (in Radiant) um die X-Achse des "aktuellen Koordinatensystems"
2. **a** : positiver Abstand entlang X zwischen zwei Gelenkachsen, angegeben in Maschineneinheiten' (mm oder Zoll), die in der INI-Datei des Systems definiert sind.
3. **d** : positive oder negative Länge entlang Z (auch in *Maschineneinheiten*)

Die Parametersätze werden immer in der gleichen Reihenfolge abgeleitet und ein Satz wird durch das Setzen des d-Parameters abgeschlossen. Dadurch bleibt die Z-Achse unseres CS nicht auf das nächste Gelenk ausgerichtet! Dies mag verwirrend erscheinen, aber wenn man sich an diese Regel hält, erhält man einen funktionierenden Satz von Parametern. Sobald der **d**-Parameter gesetzt ist, muss die X-Achse unseres CS auf die Achse des nächsten Gelenks zeigen.

9.3.5. Nummerierung der Verbindungen und Parameter

Das erste Gelenk in LinuxCNC ist Gelenk-0 (weil in der Software Zählung beginnt mit 0), während die meisten Publikationen beginnen mit der Nummer "1". Das gilt auch für alle Parameter. Das heißt, die Nummerierung beginnt mit a-0, alpha-0, d-0 und endet mit a-5, alpha-5 und d-5. Behalten Sie dies im Hinterkopf, wenn Sie einer Veröffentlichung folgen, um "genserkins"-Parameter einzurichten.

9.3.6. Wie fange ich an?

Üblicherweise wird das Referenz-CS zunächst in der Basis des Roboters platziert, wobei seine Z-Achse mit der Achse des ersten Gelenks übereinstimmt und seine X-Achse auf die Achse des nächsten Gelenks zeigt.

Dies wird auch dazu führen, dass die DRO-Werte in LinuxCNC zu diesem Punkt referenziert werden. Nachdem dies getan setzt a-0 und alpha-0 auf 0. Die oben genannten Veröffentlichung (Craig) setzt auch d-0 auf 0, die verwirrend ist, wenn eine Verschiebung Offset benötigt wird, um die Referenz-CS an der Unterseite der Basis haben. Die Einstellung von d-0 = auf die Verschiebung führt zu korrekten Ergebnissen. Auf diese Weise ist der erste Satz von Parametern alpha-0 = 0, a-0 = 0, d0 = Verschiebung, und die X-Achse des CS zeigt auf die Achse des nächsten Gelenks (Gelenk-1).

Es folgt die Ableitung der Netzmenge (alpha-1, a-1, d-1) - immer in der gleichen Reihenfolge bis hin zur sechsten Menge (alpha-5, a-5, d-5).

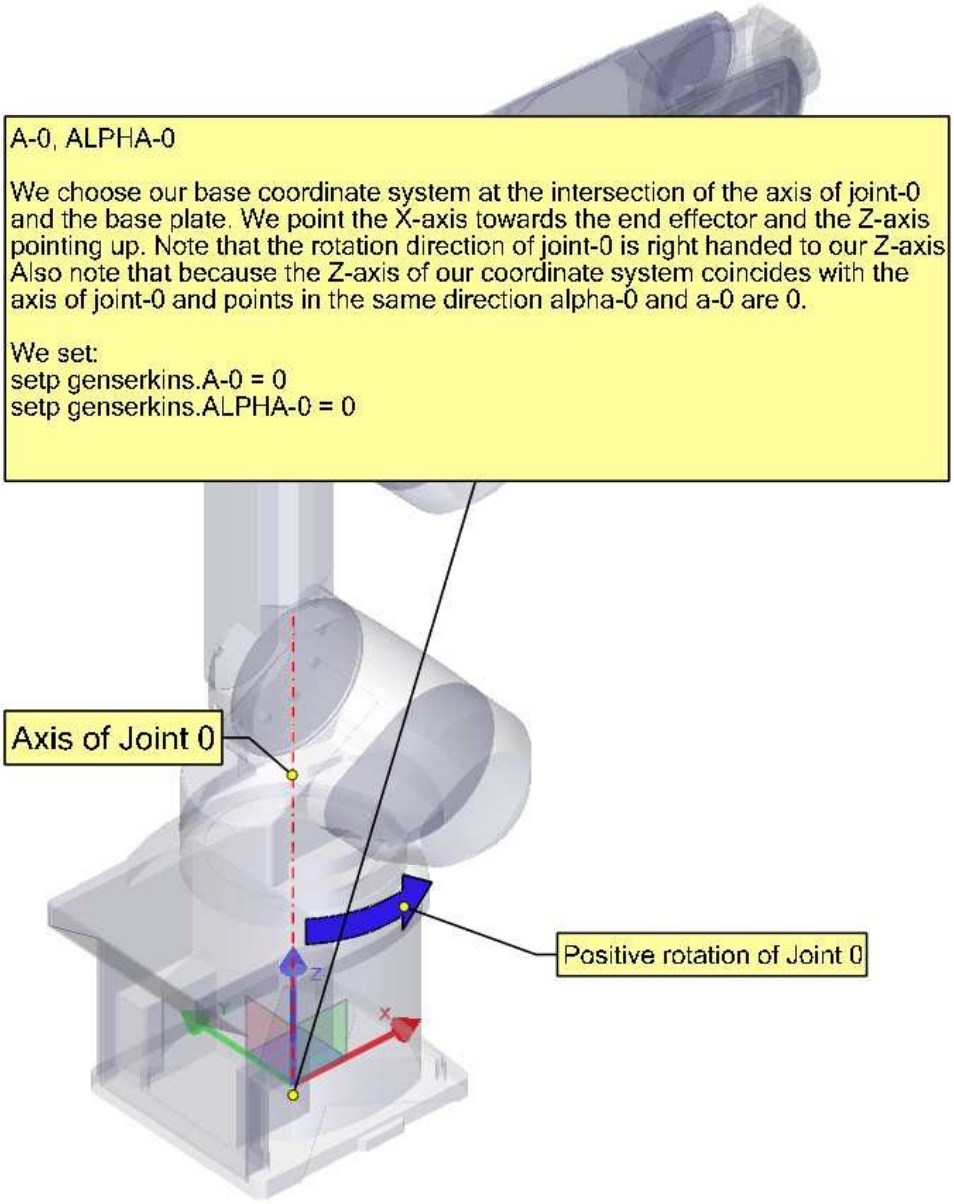
Und so sitzt der TCP-CS des Endeffektors in der Mitte des Handflansches.

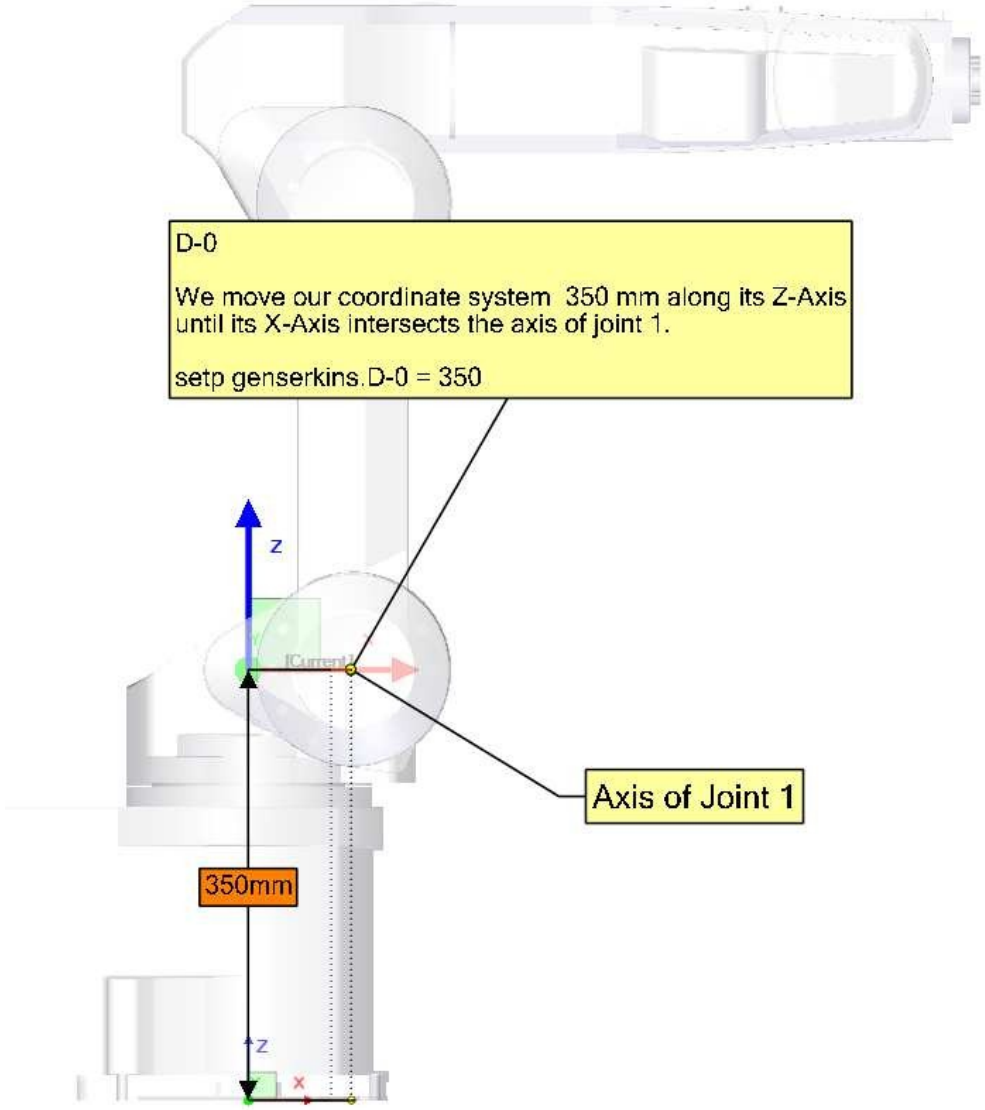
9.3.7. Sonderfälle

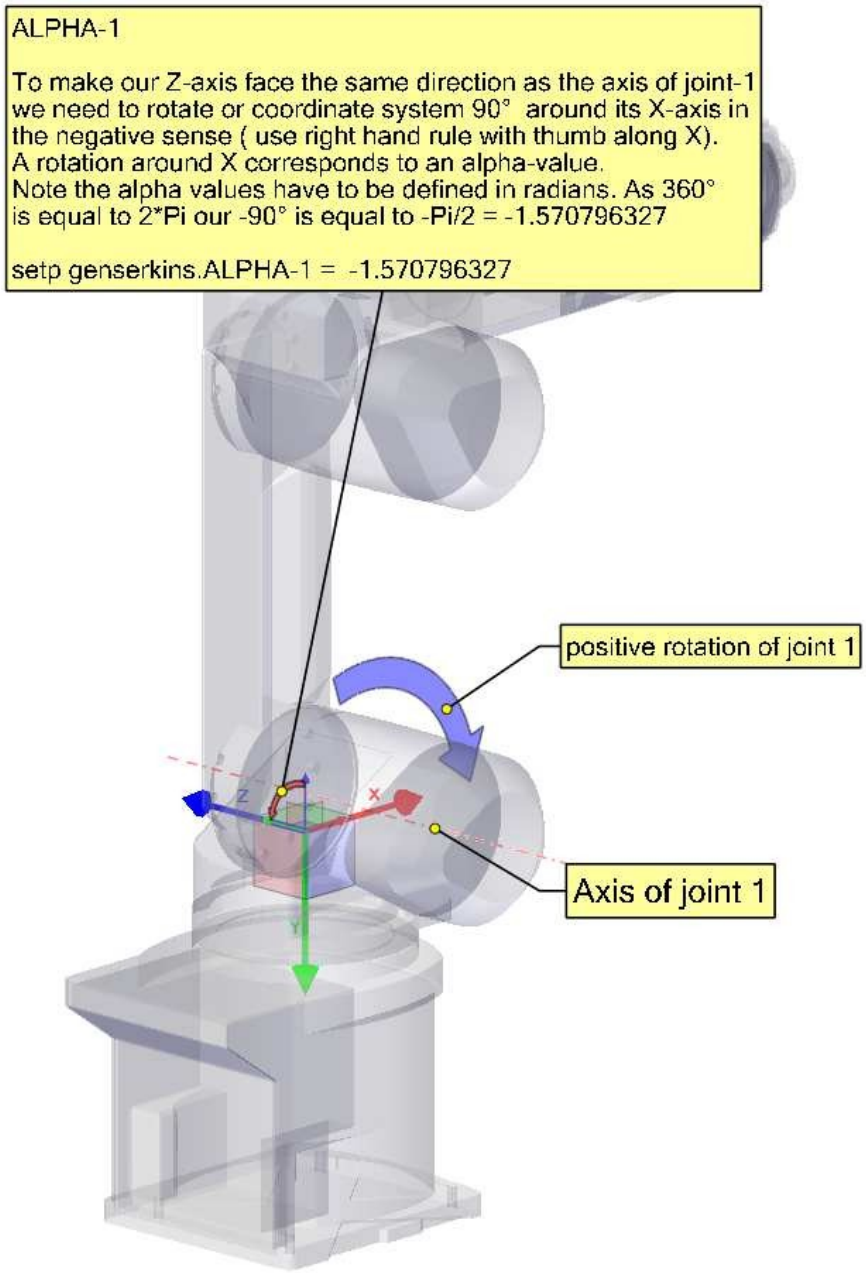
Wenn die nächste Gelenkachse parallel zur letzten ist, kann man den Wert für den d-Parameter beliebig wählen, aber es hat keinen Sinn, ihn anders als 0 zu setzen.

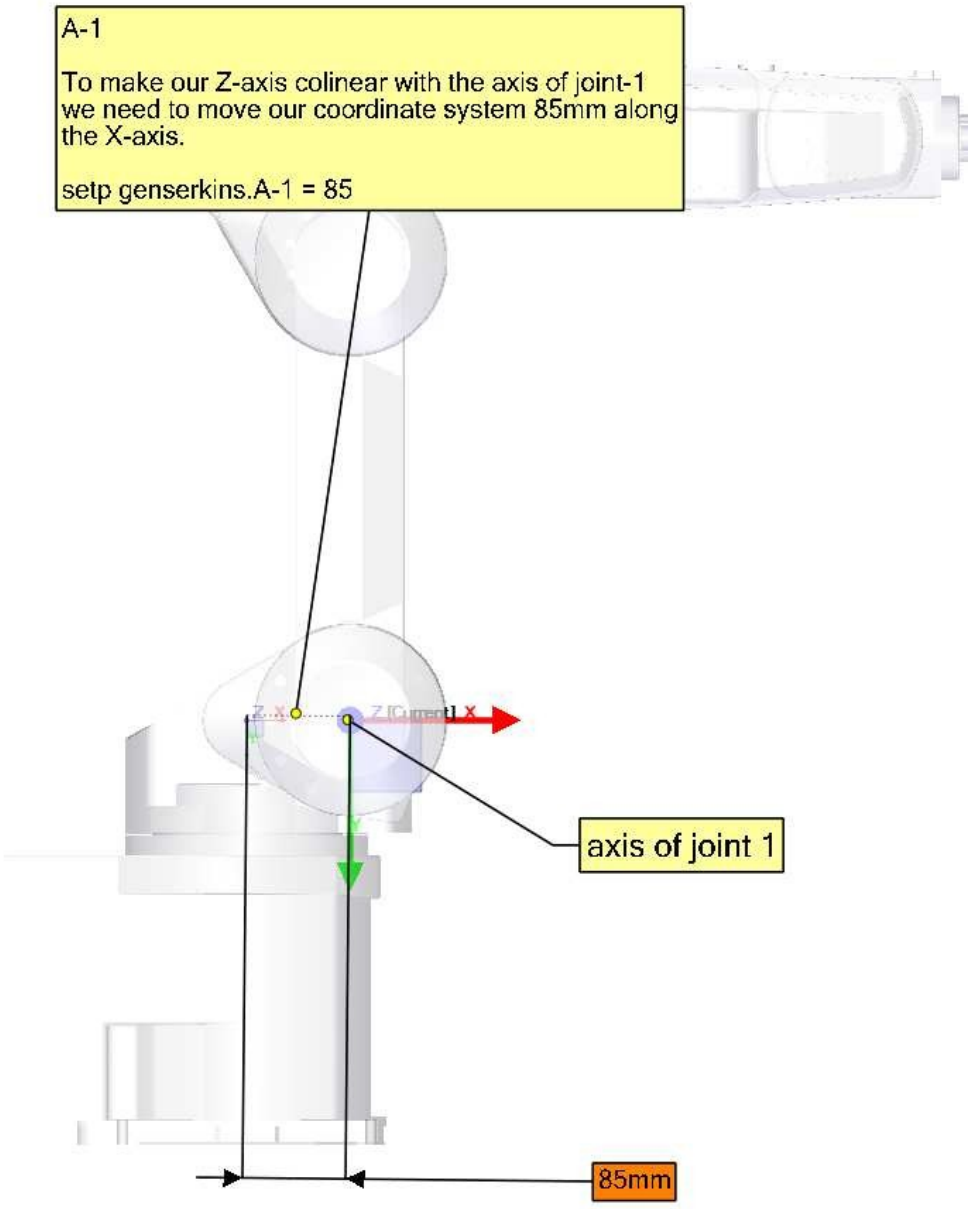
9.3.8. Detailliertes Beispiel (RV-6SL)

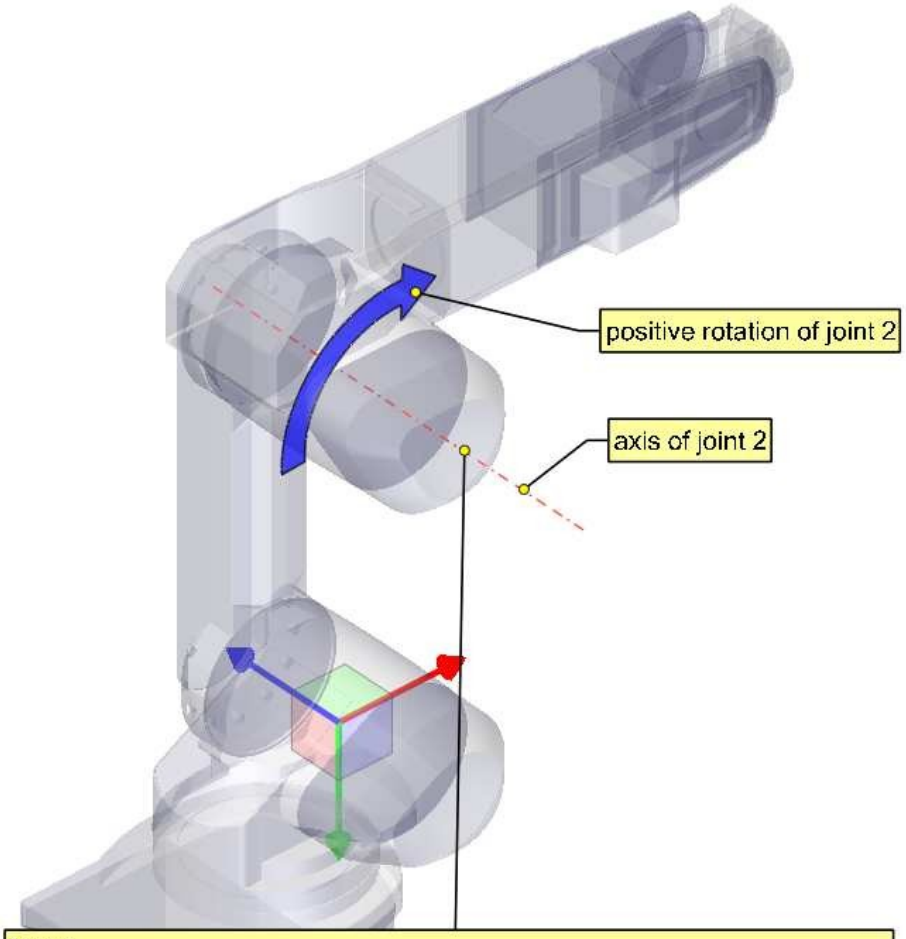
Im Folgenden wird eine Methode beschrieben, wie man die erforderlichen "modifizierten DH-Parameter" für einen Mitsubishi RV-6SDL ableitet und wie man die Parameter in der HAL-Datei einstellt, um sie mit der "genserkins"-Kinematik in LinuxCNC zu verwenden. Die erforderlichen Abmessungen werden am besten aus einer vom Hersteller des Roboters zur Verfügung gestellten Maßzeichnung entnommen.


A-0, ALPHA-0
We choose our base coordinate system at the intersection of the axis of joint-0 and the base plate. We point the X-axis towards the end effector and the Z-axis pointing up. Note that the rotation direction of joint-0 is right handed to our Z-axis Also note that because the Z-axis of our coordinate system coincides with the axis of joint-0 and points in the same direction alpha-0 and a-0 are 0.
We set:
setp genserkins.A-0 = 0
setp genserkins.ALPHA-0 = 0
Axis of Joint 0
Positive rotation of Joint 0
Z
X

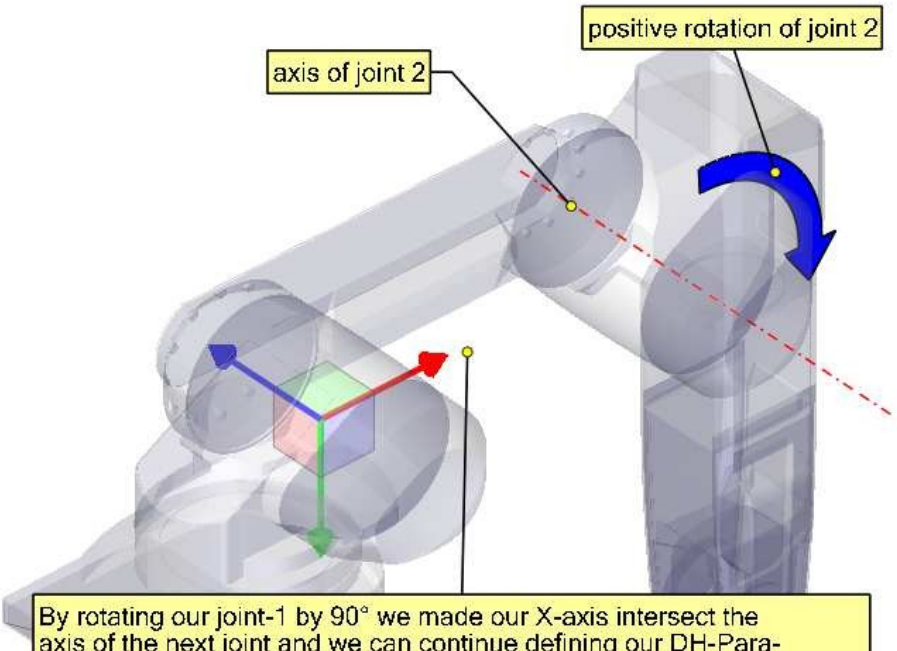

D-0
We move our coordinate system 350 mm along its Z-Axis until its X-Axis intersects the axis of joint 1.
setp genserkins.D-0 = 350
Z
Axis of Joint 1
350mm
Z
X


ALPHA-1
To make our Z-axis face the same direction as the axis of joint-1 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327
setp genserkins.ALPHA-1 = -1.570796327
positive rotation of joint 1
Axis of joint 1
Z
X
Y


A-1
To make our Z-axis colinear with the axis of joint-1 we need to move our coordinate system 85mm along the X-axis.
setp genserkins.A-1 = 85
axis of joint 1
85mm



Note:
In order to make our X-axis intersect the axis of joint-2 we would need to rotate our coordinate system around its Z-axis. To do this we could, in theory, define theta-1 equal to -90°.
However genserkins does not allow the definition of theta values. In genserkins.c we see that the theta values for all joints are set to 0.
Now theta of course is the rotation of the joint itself and so is variable in an angular joint. Theta values are only used to define the home pose of a robot in the way of an offset.
So if we could define theta-1 equal to -90° we could define joint-1 to be oriented this way for 0°. Since we cannot define it we need to rotate joint-1 in a way so that our X-axis intersects with the axis of the next joint.



By rotating our joint-1 by 90° we made our X-axis intersect the axis of the next joint and we can continue defining our DH-Parameters.

D-1

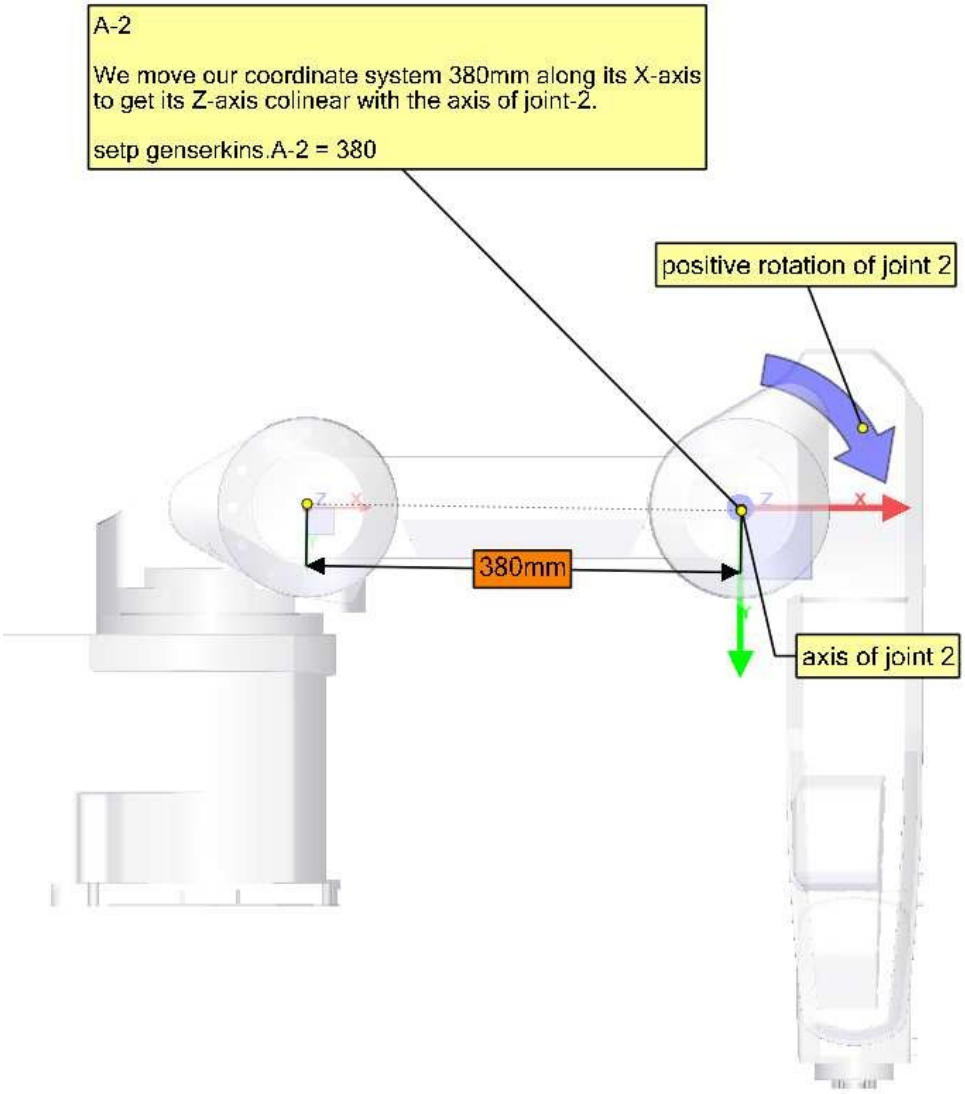
Since, after we rotated joint 1, our X-axis already intersects the axis of joint-2 we do not need to move our coordinate system along the Z-axis.

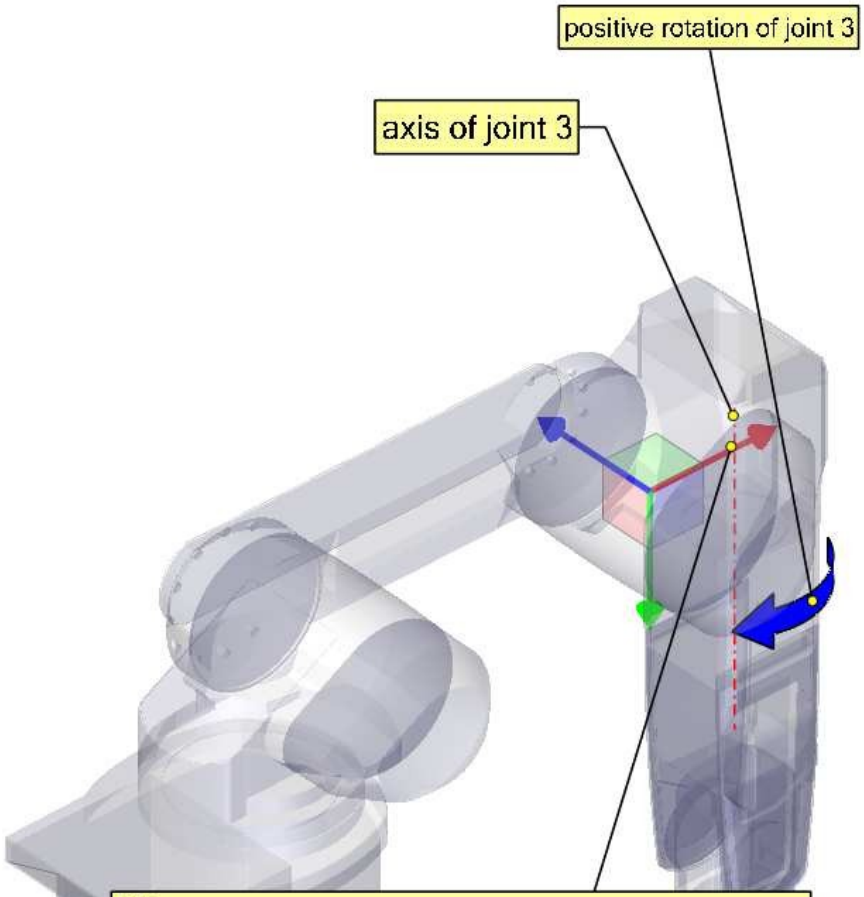
setp genserkins.D-1 = 0

ALPHA-2

The axis of joint -2 is parallel to the axis of joint -1 and points in the same direction. Thus we do not need to rotate our Z-axis.

setp genserkins.ALPHA-2 = 0


A-2
We move our coordinate system 380mm along its X-axis to get its Z-axis colinear with the axis of joint-2.
setp genserkins.A-2 = 380
positive rotation of joint 2
380mm
axis of joint 2



D-2

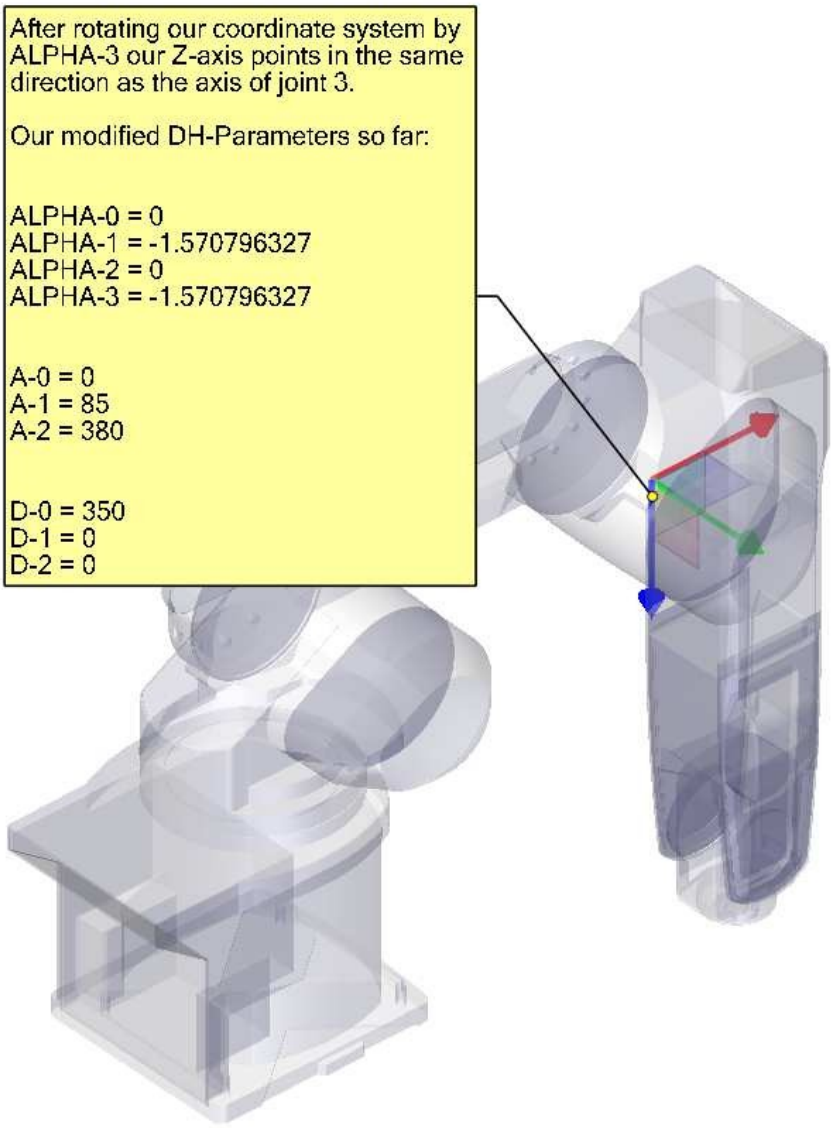
Our X-axis again already intersects the axis of the next joint 3.
So our d2 parameter is again 0.

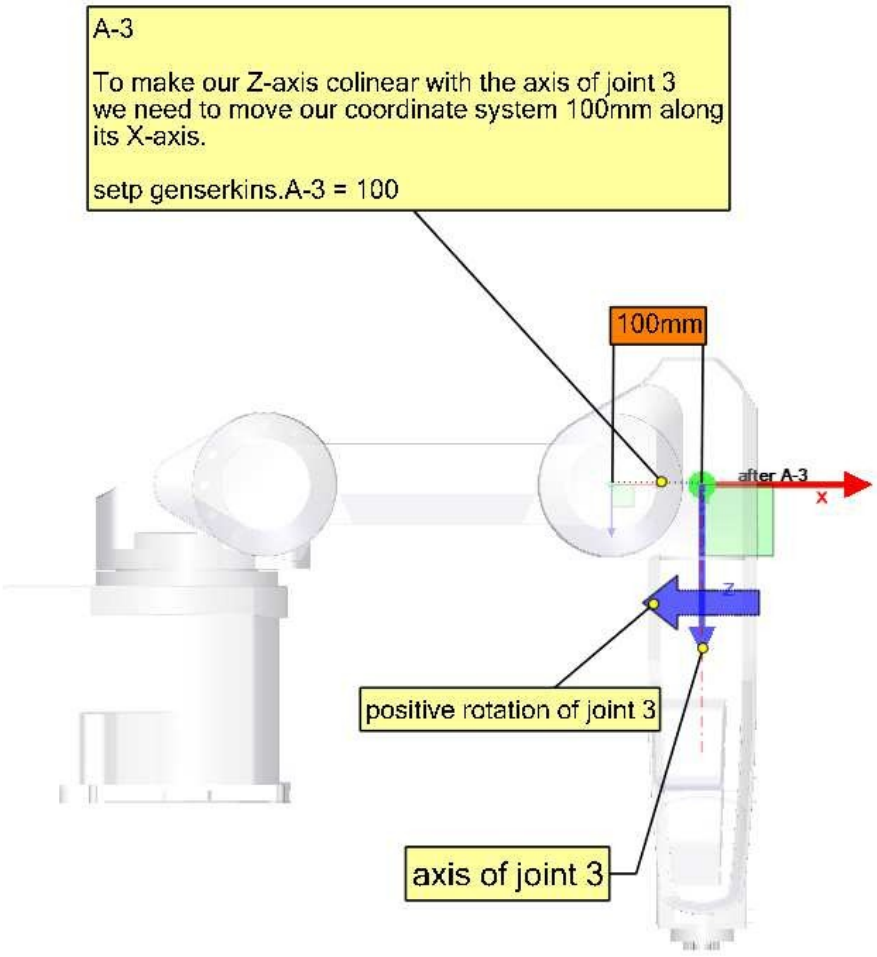
setp genserkins.D-2 = 0

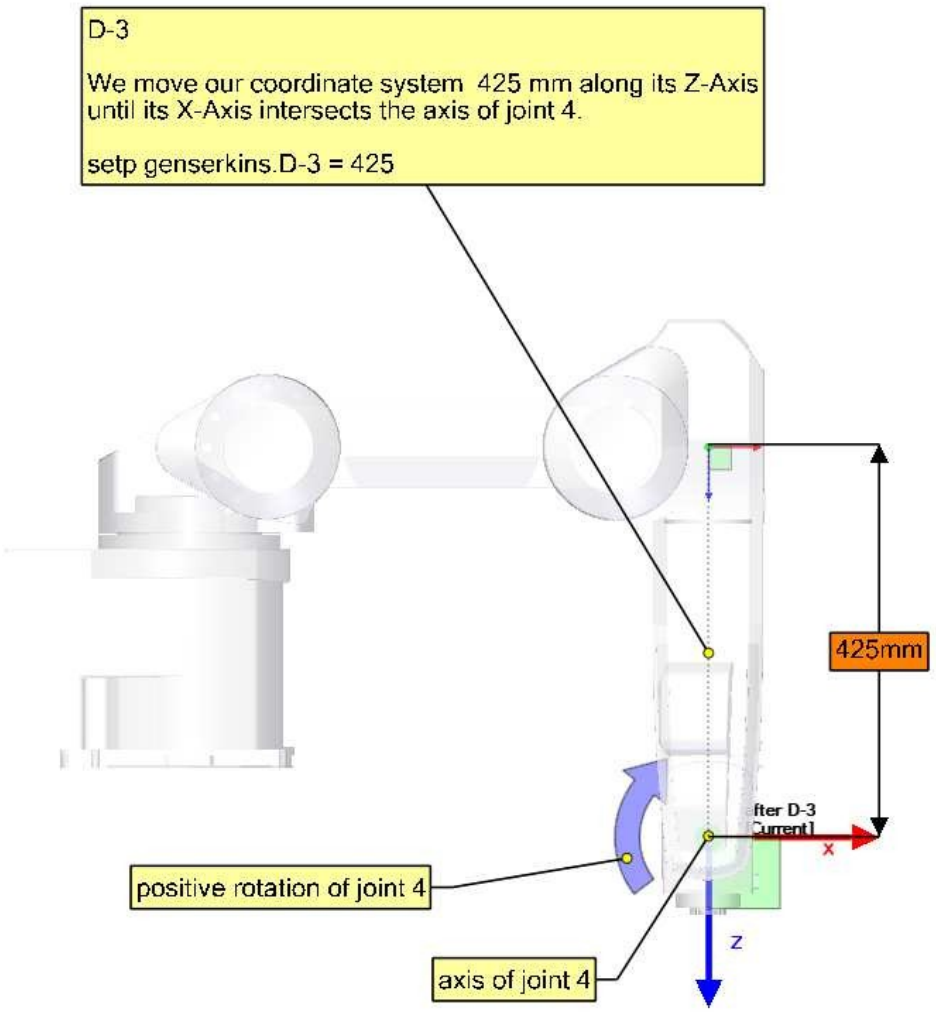
ALPHA-3

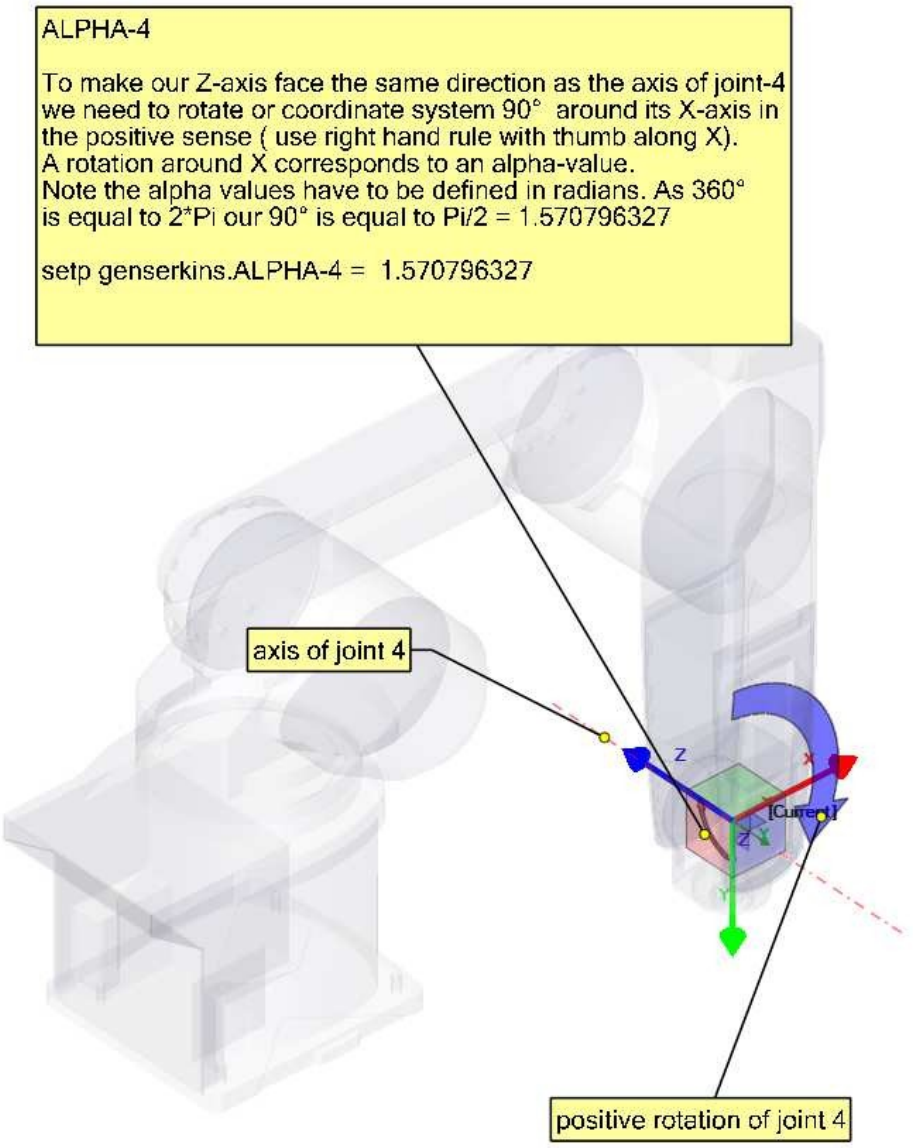
To make our Z-axis face the same direction as the axis of joint-3 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327

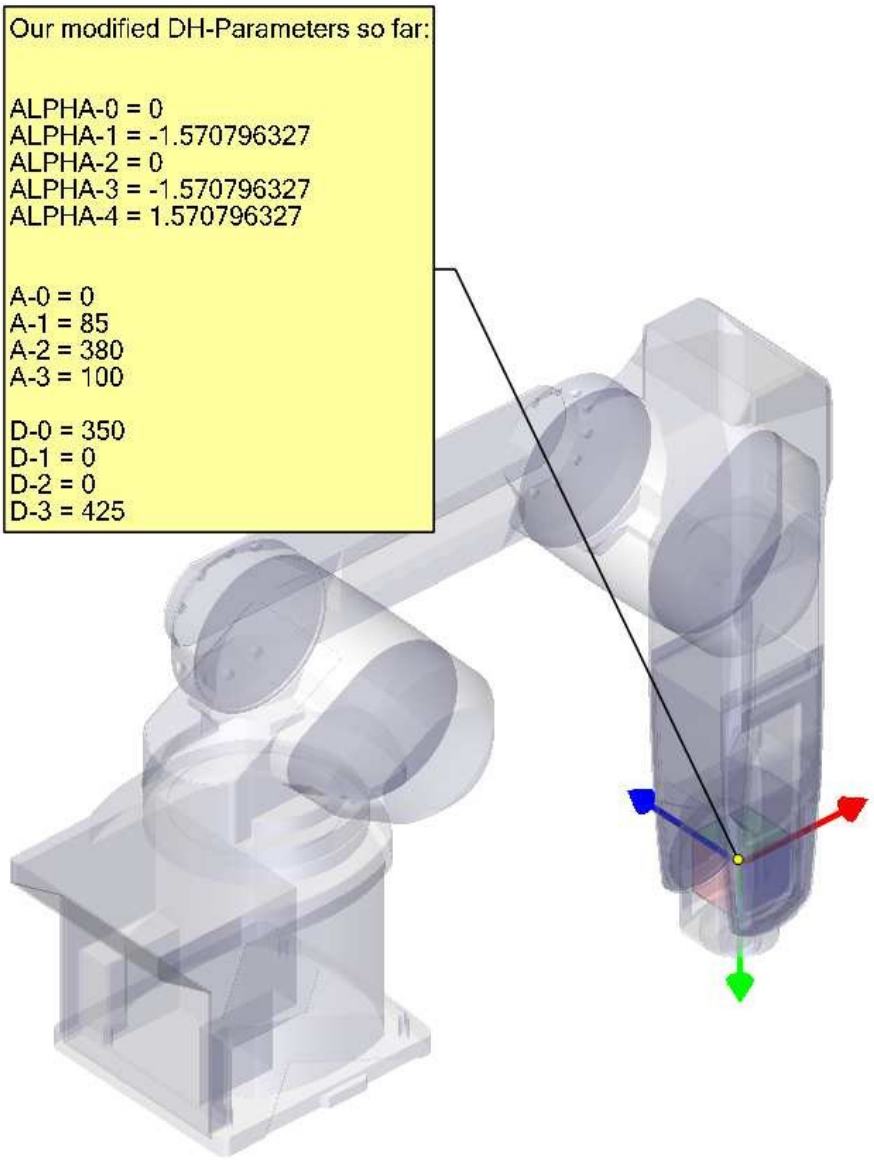
setp genserkins.ALPHA-3 = -1.570796327

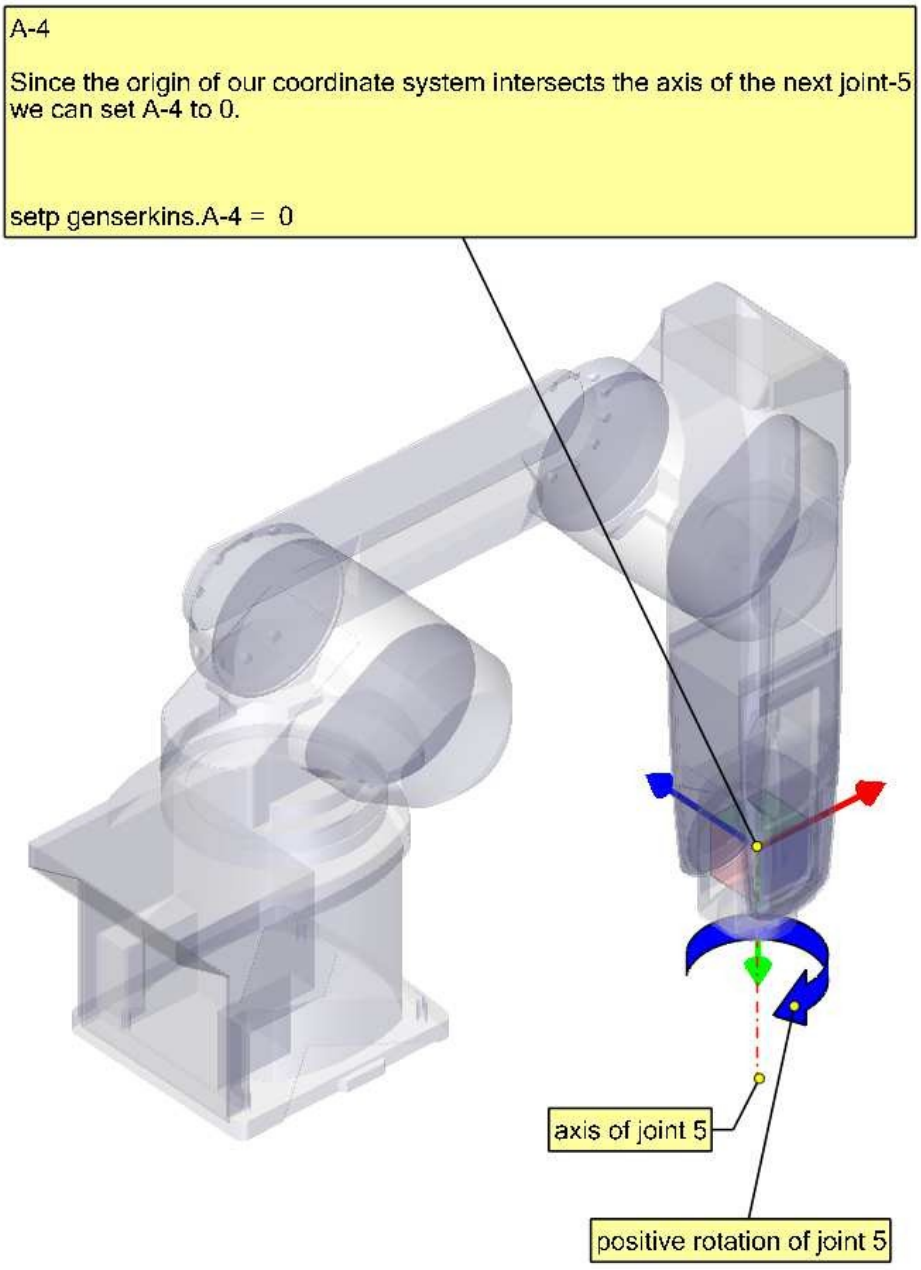

After rotating our coordinate system by ALPHA-3 our Z-axis points in the same direction as the axis of joint 3.
Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
D-0 = 350
D-1 = 0
D-2 = 0

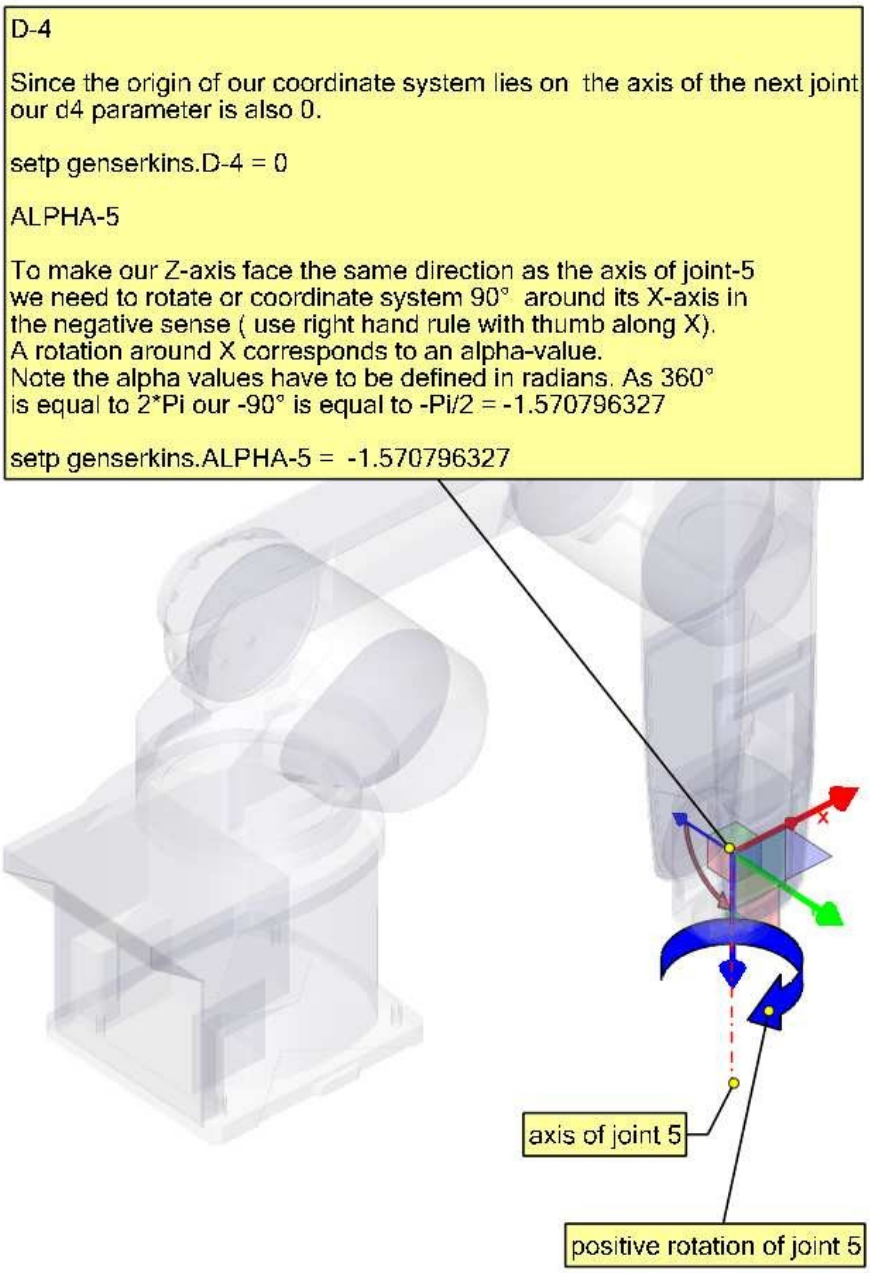

A-3
To make our Z-axis colinear with the axis of joint 3 we need to move our coordinate system 100mm along its X-axis.
setp genserkins.A-3 = 100
100mm
after A-3
X
Z
positive rotation of joint 3
axis of joint 3

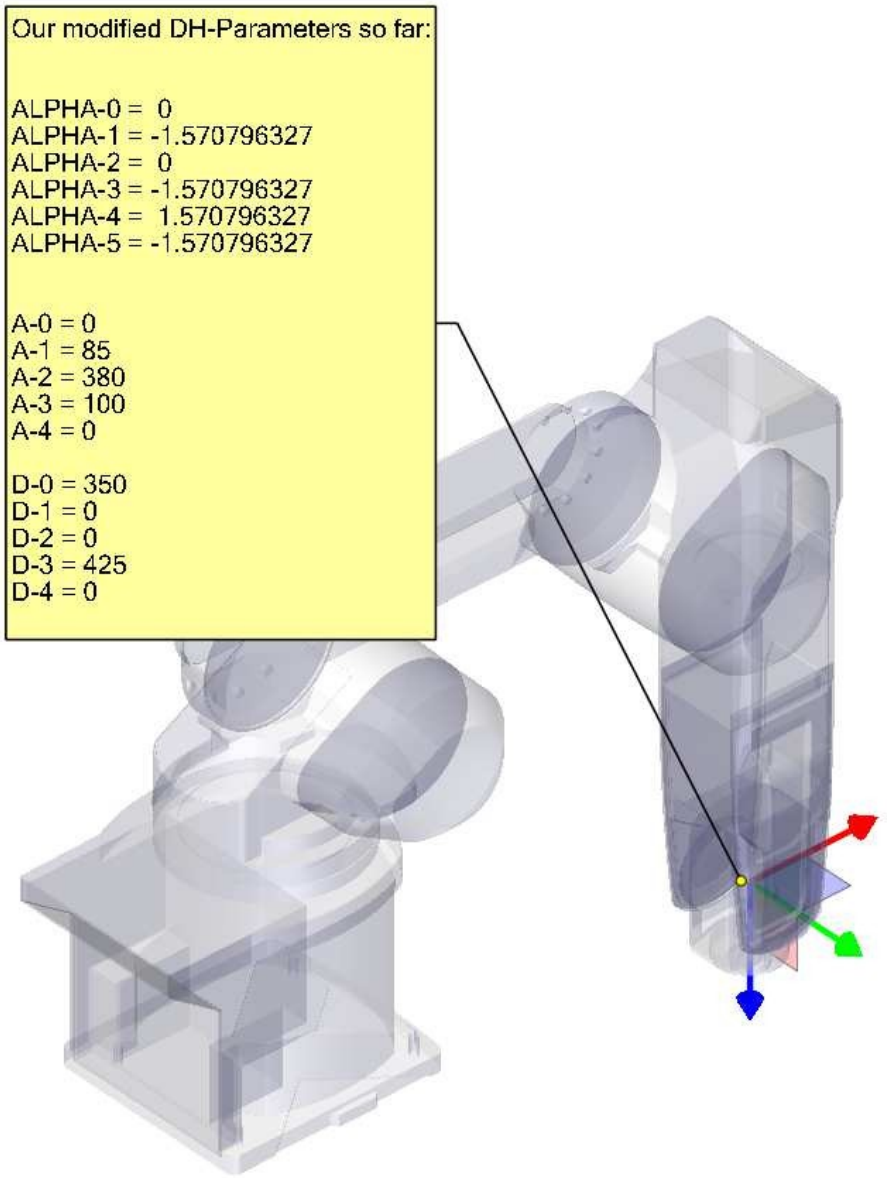

D-3
We move our coordinate system 425 mm along its Z-Axis until its X-Axis intersects the axis of joint 4.
setp genserkins.D-3 = 425
425mm
positive rotation of joint 4
axis of joint 4
x
z

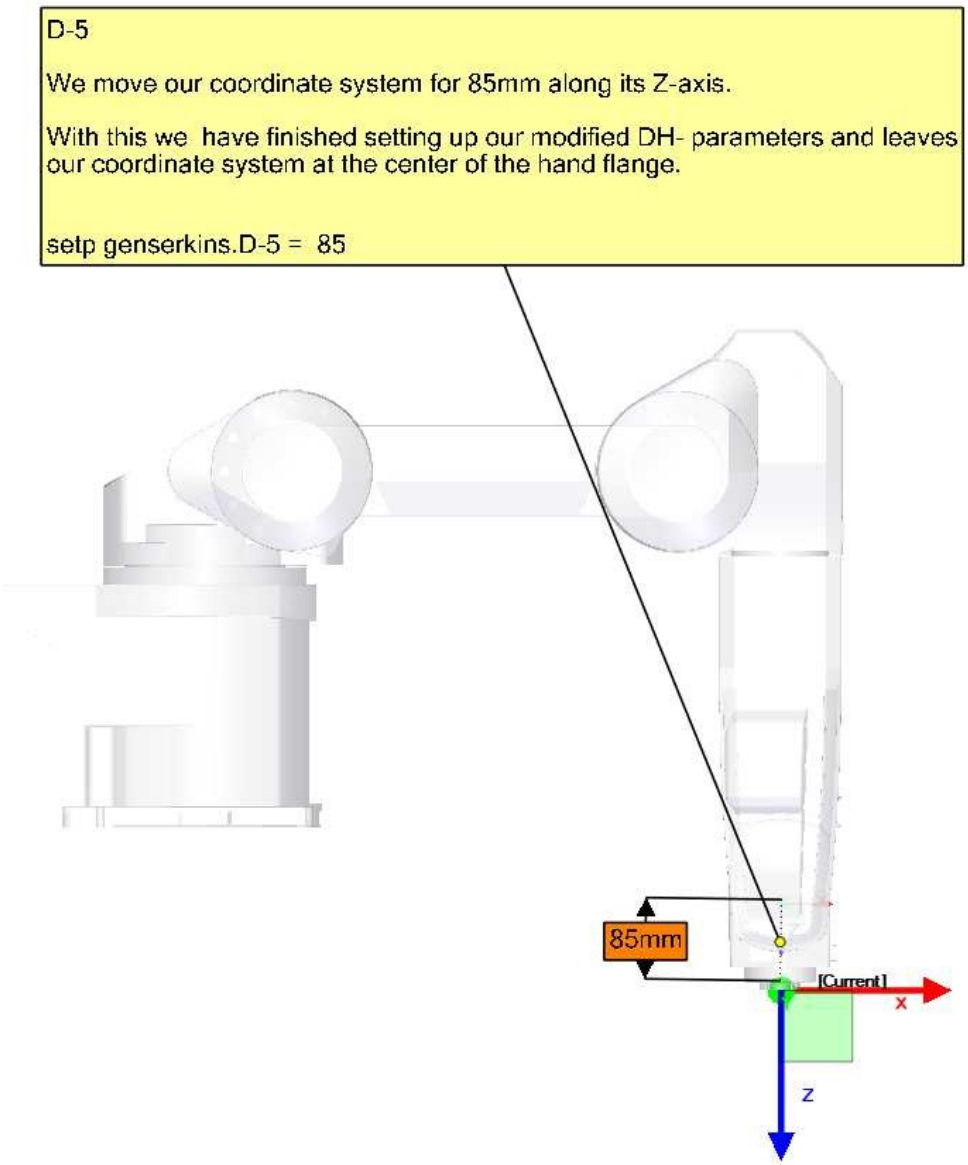

ALPHA-4
To make our Z-axis face the same direction as the axis of joint-4 we need to rotate or coordinate system 90° around its X-axis in the positive sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our 90° is equal to Pi/2 = 1.570796327
setp genserkins.ALPHA-4 = 1.570796327
axis of joint 4
positive rotation of joint 4

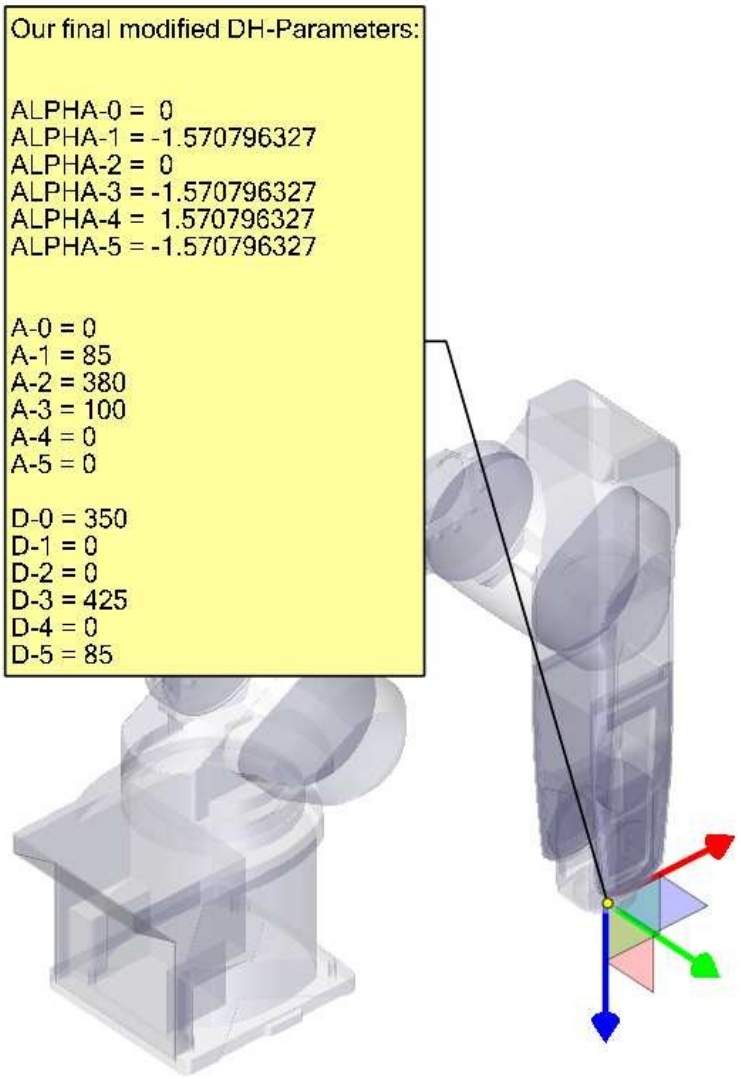

Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425


A-4
Since the origin of our coordinate system intersects the axis of the next joint-5 we can set A-4 to 0.
setp genserkins.A-4 = 0
axis of joint 5
positive rotation of joint 5


D-4
Since the origin of our coordinate system lies on the axis of the next joint our d4 parameter is also 0.
setp genserkins.D-4 = 0
ALPHA-5
To make our Z-axis face the same direction as the axis of joint-5 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327
setp genserkins.ALPHA-5 = -1.570796327
axis of joint 5
positive rotation of joint 5


Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
ALPHA-5 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
A-4 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0


D-5
We move our coordinate system for 85mm along its Z-axis.
With this we have finished setting up our modified DH- parameters and leaves our coordinate system at the center of the hand flange.
setp genserkins.D-5 = 85
85mm
[Current]
x
z



9.3.9. Danksagungen

Vielen Dank an den Benutzer Aciera für den gesamten Text und die Grafiken für den RV-6SL-Roboter!

9.4. 5-Achsen Kinematik

9.4.1. Einführung

Koordinierte mehrachsige CNC-Werkzeugmaschinen, die mit LinuxCNC gesteuert werden, erfordern

eine spezielle Kinematikkomponente für jede Art von Maschine. Dieses Kapitel beschreibt einige der gängigsten 5-Achsen-Maschinenkonfigurationen und entwickelt dann die Vorwärts- (von Arbeits- zu Gelenkkoordinaten) und Rückwärtstransformationen (von Gelenk zu Arbeit) in einem allgemeinen mathematischen Prozess für zwei Maschinentypen.

Die kinematischen Komponenten werden ebenso dargestellt wie Vismach-Simulationsmodelle, um ihr Verhalten auf dem Computerbildschirm zu demonstrieren. Es werden auch Beispiele für HAL-Dateien gegeben.

Beachten Sie, dass sich bei dieser Kinematik die Rotationsachsen in die entgegengesetzte Richtung bewegen, wie es die Konvention vorsieht. Siehe Abschnitt ["Rotationsachsen"] (https://linuxcnc.org/docs/html/gcode/machining-center.html#_rotational_axes) für Details.

9.4.2. 5-Achsen-Werkzeugmaschinen-Konfigurationen

In diesem Abschnitt befassen wir uns mit den typischen 5-Achsen-Fräs- oder Oberfräsmaschinen mit fünf Gelenken oder Freiheitsgraden, die in koordinierten Bewegungen gesteuert werden.

3-Achsen-Werkzeugmaschinen können die Werkzeugausrichtung nicht ändern, daher verwenden 5-Achsen-Werkzeugmaschinen zwei zusätzliche Achsen, um das Schneidwerkzeug in eine geeignete Ausrichtung für die effiziente Bearbeitung von Freiformflächen zu bringen.

Typische Konfigurationen von 5-Achsen-Werkzeugmaschinen sind in den Abbildungen 3, 5, 7 und 9-11 [1,2] im Abschnitt Abbildungen dargestellt.

Die Kinematik von 5-Achsen-Werkzeugmaschinen ist viel einfacher als die von 6-Achsen-Serienarmrobotern, da 3 der Achsen normalerweise lineare Achsen und nur zwei rotierende Achsen sind.

9.4.3. Werkzeugausrichtung und -position

CAD/CAM-Systeme werden in der Regel verwendet, um die 3D-CAD-Modelle des Werkstücks sowie die CAM-Daten für die Eingabe in die CNC-5-Achsen-Maschine zu erzeugen. Die Daten zur Werkzeug- oder Fräserposition (CL) setzen sich aus der Position der Fräserspitze und der Ausrichtung des Fräsers relativ zum Werkstückkoordinatensystem zusammen. Zwei Vektoren, wie sie von den meisten CAM-Systemen erzeugt werden und in Abb. 1 dargestellt sind, enthalten diese Informationen:

$$K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} \text{ orientation vector;} \qquad Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \text{ position vector} \qquad (1)$$

Der K-Vektor entspricht dem dritten Vektor der Pose-Matrix E_6, die in der 6-Achsen-Roboterkinematik [3] verwendet wurde, und der Q-Vektor entspricht dem vierten Vektor von E_6. Vektor von E_6. In MASTERCAM zum Beispiel sind diese Informationen in der Zwischenausgabedatei ".nci" enthalten.

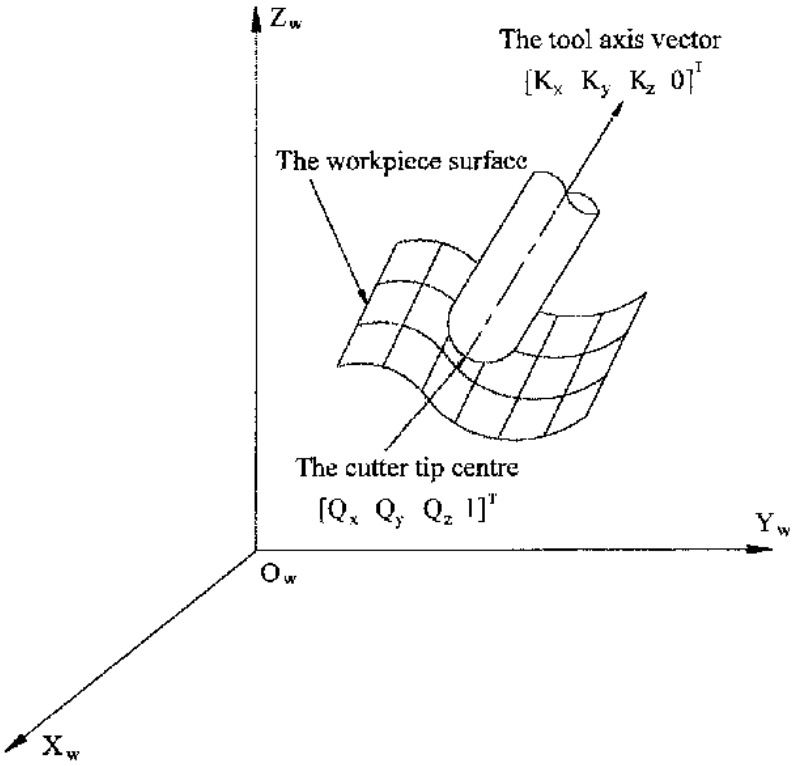


Figure 138. Standortdaten des Fräsers

9.4.4. Translations- und Rotationsmatrizen

Homogene Transformationen bieten eine einfache Möglichkeit, die Mathematik der Mehrachsenkinematik von Maschinen zu beschreiben. Eine Transformation des Raums H ist eine 4x4-Matrix und kann Translations- und Rotationstransformationen darstellen. Wird ein Punkt x,y,x durch einen Vektor u = {x,y,z,1}^{T} beschrieben, so wird seine Transformation v durch das Matrixprodukt

$$v = H \cdot u$$

Es gibt vier grundlegende Transformationsmatrizen, auf die sich die 5-Achsen-Kinematik stützen kann:

$$T(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

$$R(Y,\theta) = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(Z,\theta) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

Die Matrix T(a,b,c) impliziert eine Verschiebung in den Koordinatenrichtungen X, Y und Z um die Beträge a, b bzw. c. Die R-Matrizen implizieren Rotationen des Winkels theta um die X-, Y- bzw. Z-Koordinatenachse. Die Symbole "C" und "S" beziehen sich auf die Kosinus- bzw. Sinusfunktionen.

9.4.5. Tisch Dreh-/Schwenkkonfigurationen mit 5 Achsen (engl. Table Rotary/Tilting 5-Axis Configurations)

Bei diesen Werkzeugmaschinen sind die beiden Rotationsachsen auf dem Arbeitstisch der Maschine montiert. Typischerweise werden zwei Formen verwendet:

- Ein Drehtisch, der sich um die vertikale Z-Achse dreht (C-Drehung, sekundär), ist auf einem Kipptisch montiert, der sich um die X- oder Y-Achse dreht (A- oder B-Drehung, primär). Das Werkstück ist auf dem Drehtisch montiert.
- Ein Kipptisch, der sich um die X- oder Y-Achse dreht (A- oder B-Drehung, sekundär), ist auf einem Drehtisch montiert, der sich um die Z-Achse dreht (C-Drehung, primär), wobei das Werkstück auf dem Kipptisch liegt.

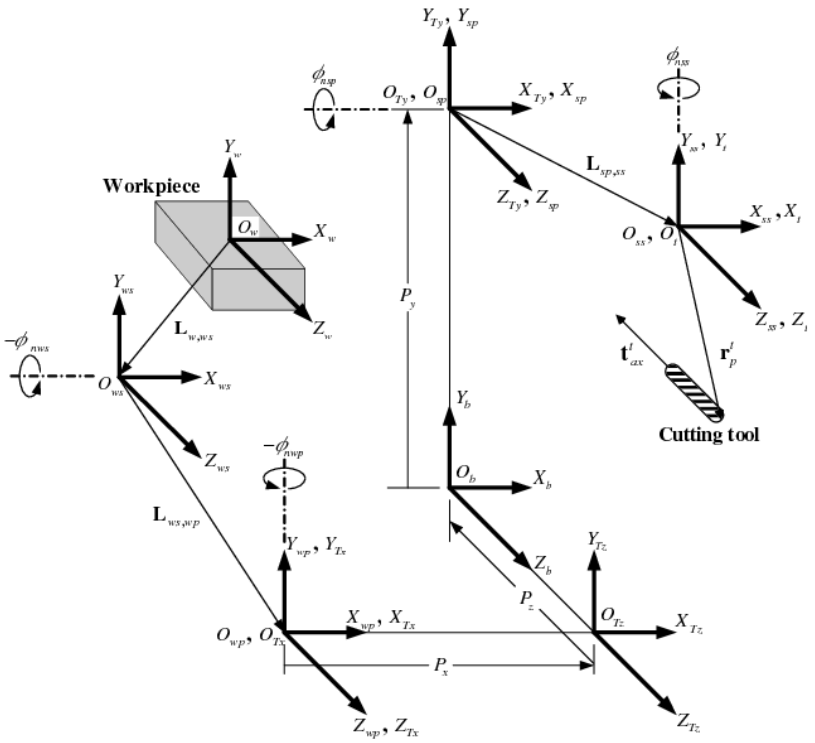


Figure 139. Allgemeine Konfiguration und Koordinatensysteme

Eine mehrachsige Maschine kann als eine Reihe von Gliedern betrachtet werden, die durch Gelenke verbunden sind. Durch die Einbettung eines Koordinatenrahmens in jedes Glied der Maschine und die Verwendung homogener Transformationen können wir die relative Position und Orientierung zwischen diesen Koordinatenrahmen beschreiben

Wir müssen eine Beziehung zwischen dem Werkstückkoordinatensystem und dem Werkzeugkoordinatensystem beschreiben. Dies kann durch eine Transformationsmatrix "${}^{w}A_{t}$" definiert werden, die durch nachfolgende Transformationen zwischen den verschiedenen Strukturelementen oder Gliedern der Maschine, die jeweils ihr eigenes definiertes Koordinatensystem haben, gefunden werden kann. Im Allgemeinen kann eine solche Transformation wie folgt aussehen:

$$ {}^{w}A_{t} = {}^{w}A_{1} \cdot {}^{1}A_{2} \cdot {}^{2}A_{3} \cdots {}^{n}A_{t} \qquad (4) $$

wobei jede Matrix ${}^{i-1}A_{j}$ eine Translationsmatrix *T* oder eine Rotationsmatrix *R* der Form (2,3) ist.

Die Matrixmultiplikation ist ein einfacher Vorgang, bei dem die Elemente jeder Zeile der linken Matrix A mit den Elementen jeder Spalte der rechten Matrix B multipliziert und summiert werden, um ein Element der Ergebnismatrix C zu erhalten.

$$C_{ij} = \sum_{k=1,n}^{n} A_{ik} B_{kj}; \quad i = 1, n; \quad j = 1, n$$

In Abb. 2 ist eine generische Konfiguration mit Koordinatensystemen dargestellt [4]. Sie umfasst sowohl Tischdreh-/Schwenkachsen als auch Spindel-Dreh-/Schwenkachsen. Nur zwei der Drehachsen werden tatsächlich in einer Werkzeugmaschine verwendet.

Zunächst werden wir die Transformationen für die erste der oben erwähnten Konfigurationen entwickeln, d. h. einen Tisch vom Typ Kippen/Drehen (trt) ohne Drehachsenversatz. Wir können ihr den Namen xyzac-trt-Konfiguration geben.

Wir entwickeln auch die Transformationen für den gleichen Typ (xyzac-trt), aber mit rotierenden Achsenversätzen.

Dann entwickeln wir die Transformationen für eine xyzbc-trt-Konfiguration mit Rotationsachsen-Offsets.

Transformationen für eine xyzac-trt-Werkzeugmaschine mit Werkstückversatz

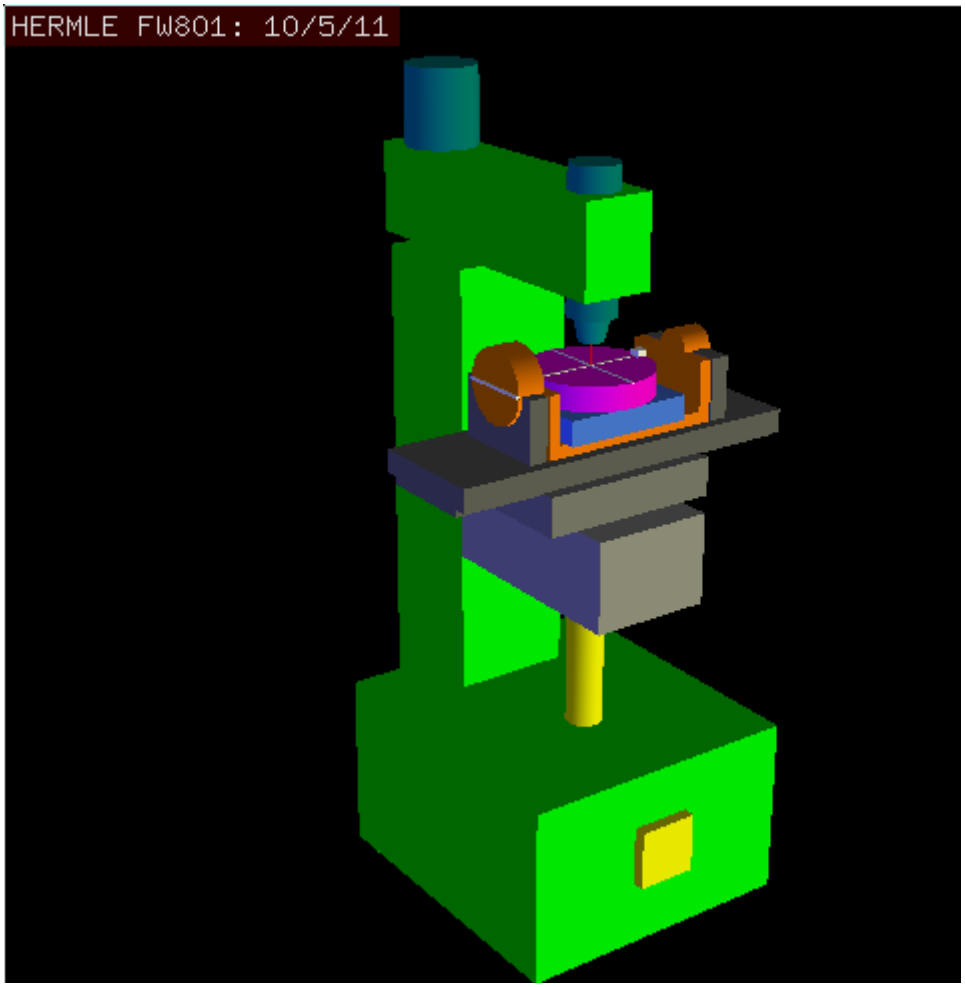


Figure 140. vismach-Modell von xyzac-trt mit übereinstimmenden Drehachsen

Wir befassen uns hier mit einer vereinfachten Konfiguration, bei der sich die Kippachse und die Drehachse in einem Punkt schneiden, der als Drehpunkt bezeichnet wird, wie in Abb. 4 dargestellt.

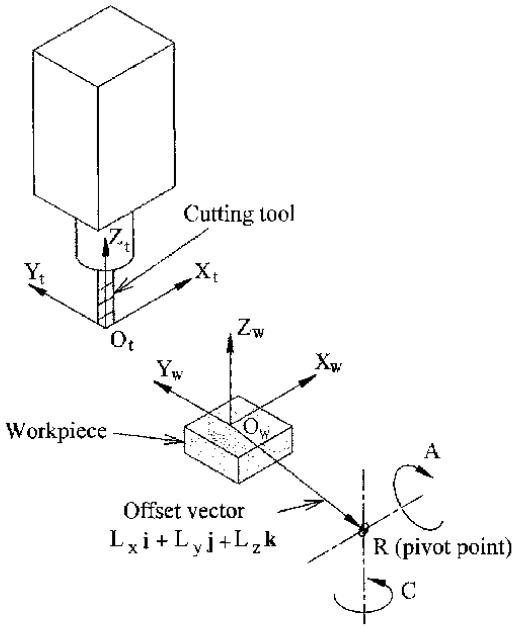


Figure 141. Kipp-/Drehkonfiguration des Tisches

Vorwärts-Transformation

Die Transformation kann durch die sequentielle Multiplikation der Matrizen definiert werden:

$$ {}^{w}A_{t} = {}^{w}A_{C} \cdot {}^{C}A_{A} \cdot {}^{A}A_{P} \cdot {}^{P}A_{t} \qquad (5) $$

wobei die Matrizen wie folgt aufgebaut sind:

$$ {}^{w}A_{C} = \begin{bmatrix} 1 & 0 & 0 & L_x \\ 0 & 1 & 0 & L_y \\ 0 & 0 & 1 & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{C}A_{A} = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6) $$

$$ {}^{A}A_{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_{t} = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7) $$

In diesen Gleichungen definieren L_x, L_y, L_z die Verschiebungen des Drehpunktes der beiden Drehachsen A und C relativ zum Ursprung des Werkstückkoordinatensystems. Außerdem sind P_x, P_y, P_z die relativen Abstände des Drehpunkts zur Position der Fräserspitze, die auch als "Gelenkkoordinaten" des Drehpunkts bezeichnet werden können. Der Drehpunkt liegt im Schnittpunkt der beiden Drehachsen. Die Vorzeichen der Terme S_A und S_C unterscheiden sich von denen in [2,3], da dort die Tischdrehungen relativ zu den Werkstückkoordinatenachsen negativ sind (beachten Sie, dass sin(-theta) = -sin(theta), cos(-theta) = cos(theta)).

Multipliziert mit (5) ergibt sich das Ergebnis:

$$ {}^{w}A_{t} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ 0 & -S_A & C_A & -S_A P_y + C_A P_z + L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (8) $$

Wir können nun die dritte Spalte dieser Matrix mit unserem gegebenen Werkzeugorientierungsvektor K gleichsetzen, d. h.:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (9) $$

Aus diesen Gleichungen lassen sich die Drehwinkel $theta_A$, $theta_C$ ermitteln. Aus der dritten Zeile finden wir:

$$ \theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (10) $$

und durch Division der ersten Zeile durch die zweite Zeile ergibt sich:

$$ \theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (11) $$

Diese Beziehungen werden normalerweise im CAM-Postprozessor verwendet, um die Vektoren der Werkzeugausrichtung in Drehwinkel umzuwandeln.

Indem wir die letzte Spalte von (8) mit dem Werkzeugpositionsvektor Q gleichsetzen, können wir schreiben:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ -S_A P_y + C_A P_z + L_z \\ 1 \end{bmatrix} \qquad (12) $$

Der Vektor auf der rechten Seite kann auch als das Produkt einer Matrix und eines Vektors geschrieben werden, was folgendes ergibt:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & L_x \\ -S_C & C_C C_A & C_C S_A & L_y \\ 0 & -S_A & C_A & L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (13) $$

Dies kann wie folgt erweitert werden

$$ \begin{aligned} Q_x &= C_c P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ Q_y &= -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ Q_z &= -S_A P_y + C_A P_z + L_z \end{aligned} \qquad (14) $$

was die Vorwärtstransformation der Kinematik darstellt.

Inverse Transformation

Wir können P aus Gleichung (13) als "P = $({}^{Q}A_{P})^{-1}$ * Q" berechnen. Die quadratische Matrix ist eine homogene 4x4-Matrix, die eine Rotationsmatrix R und einen Translationsvektor q enthält, deren

Umkehrung wie folgt geschrieben werden kann:

$$ {}^{q}A_{p} = \begin{bmatrix} R & q \\ 0 & 1 \end{bmatrix} \qquad ({}^{Q}A_{P})^{-1} = \begin{bmatrix} R^{T} & -R^{T}q \\ 0 & 1 \end{bmatrix} \qquad (15) $$

wobei R^T die Transponierung von R ist (Zeilen und Spalten vertauscht). Wir erhalten also:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & -C_C L_x + S_C L_y \\ S_C C_A & C_C C_A & -S_A & -S_C C_A L_x - C_C C_A L_y + S_A L_z \\ S_C S_A & C_C S_A & C_A & -S_C S_A L_x - C_C S_A L_y - C_A L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (16) $$

Die gewünschten Gleichungen für die *inverse Transformation* der Kinematik können somit wie folgt geschrieben werden:

$$ \begin{aligned} P_x &= C_c(Q_x - L_x) - S_C(Q_y - L_y) \\ P_y &= S_C C_A(Q_x - L_x) + C_C C_A(Q_y - L_y) - S_A(Q_z - L_z) \\ P_z &= S_C S_A(Q_x - L_x) + C_C S_A(Q_y - L_y) + C_A(Q_z - L_z) \end{aligned} \qquad (17) $$

Transformationen für eine xyzac-trt-Maschine mit Drehachsenverschiebungen

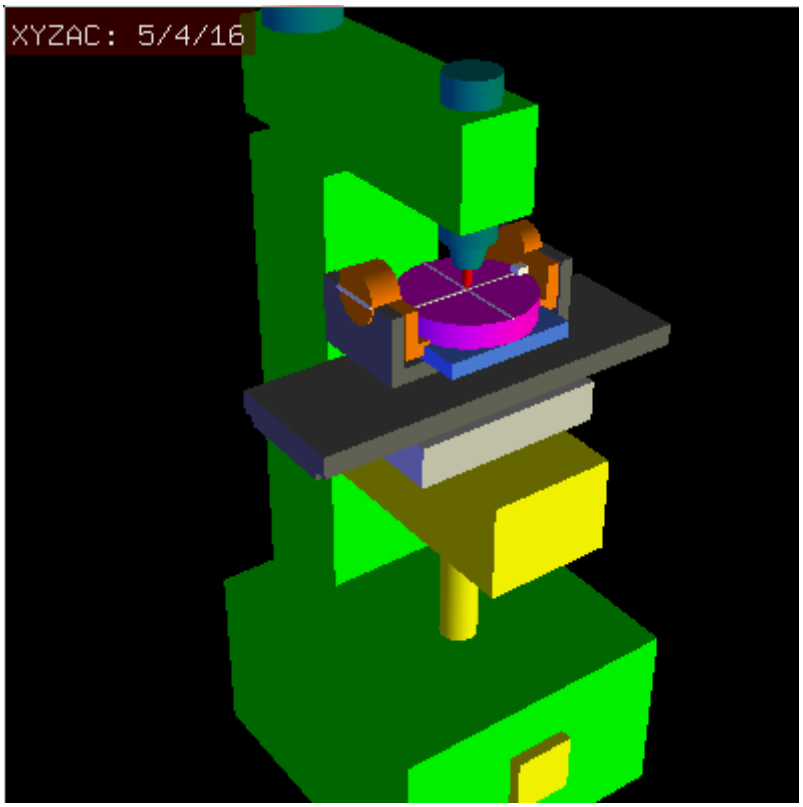


Figure 142. Vismach-Modell von xyzac-trt mit Rotationsachsenversatz (positiv)

Wir haben es hier mit einer erweiterten Konfiguration zu tun, bei der sich die Kippachse und die Drehachse nicht in einem Punkt schneiden, sondern einen Versatz D_y aufweisen. Außerdem gibt es zwischen den beiden Koordinatensystemen O_{ws} und O_{wp} aus Abb. 2 einen z-Versatz, der D_z genannt wird. Ein Vismach-Modell ist in Abb. 5 dargestellt, und die Offsets sind in Abb. 6 gezeigt (positive Offsets in diesem Beispiel). Um die Konfiguration zu vereinfachen, werden die Versätze L_x, L_y, L_z des vorherigen Falls nicht berücksichtigt. Sie sind wahrscheinlich nicht notwendig, wenn man die G54 Offsets in LinuxCNC mit Hilfe der "touch of"-Funktion verwendet.

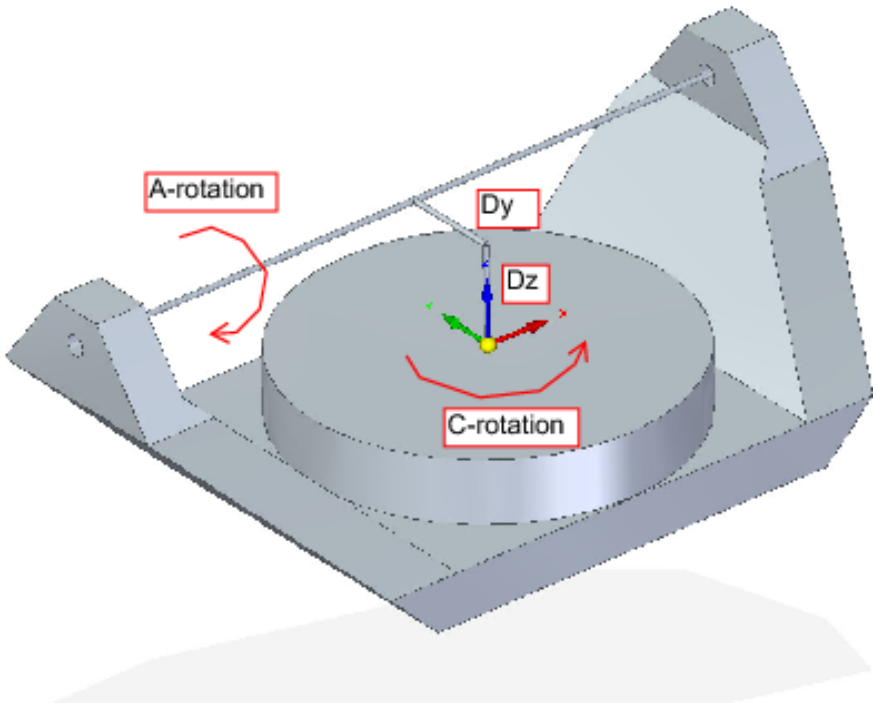


Figure 143. Kipp-/Drehkonfiguration des Tisches xyzac-trt, mit Achsenversatz

Vorwärts-Transformation

Die Transformation kann durch die sequentielle Multiplikation der Matrizen definiert werden:

$$ {}^{w}A_{t} = {}^{w}A_{O} \cdot {}^{O}A_{A} \cdot {}^{A}A_{P} \cdot {}^{P}A_{t} \qquad (18) $$

wobei die Matrizen wie folgt aufgebaut sind:

$$ {}^{w}A_{O} = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & D_y \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (19) $$

$$ {}^{A}A_{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_{t} = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y - D_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (20) $$

In diesen Gleichungen definieren D_y, D_z die Verschiebungen des Drehpunktes der Drehachsen A relativ zum Ursprung des Werkstückkoordinatensystems. Außerdem sind P_x, P_y, P_z die relativen Abstände des Drehpunkts zur Position der Schneidenspitze, die auch als "Gelenkkoordinaten" des Drehpunkts bezeichnet werden können. Der Drehpunkt liegt auf der Drehachse A.

Bei Multiplikation gemäß (18) erhalten wir:

$$ {}^{w}A_{t} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A (P_y - D_y) + S_C S_A (P_z - D_z) + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A (P_y - D_y) + C_C S_A (P_z - D_z) + C_C D_y \\ 0 & -S_A & C_A & -S_A (P_y - D_y) + C_A (P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (21) $$

Wir können nun die dritte Spalte dieser Matrix mit unserem gegebenen Werkzeugorientierungsvektor K gleichsetzen, d. h.:

$$K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (22)$$

Aus diesen Gleichungen lassen sich die Drehwinkel theta$_A$, theta$_C$ ermitteln. Aus der dritten Zeile finden wir:

$$\theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (23)$$

und durch Division der zweiten Zeile durch die erste Zeile ergibt sich:

$$\theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (24)$$

Diese Beziehungen werden normalerweise im CAM-Postprozessor verwendet, um die Vektoren der Werkzeugausrichtung in Drehwinkel umzuwandeln.

Wenn wir die letzte Spalte von (21) mit dem Werkzeugpositionsvektor Q gleichsetzen, können wir schreiben:

$$Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A (P_y - D_y) + S_C S_A (P_z - D_z) + S_C D_y \\ -S_C P_x + C_C C_A (P_y - D_y) + C_C S_A (P_z - D_z) + C_C D_y \\ -S_A (P_y - D_y) + C_A (P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (25)$$

Der Vektor auf der rechten Seite kann auch als das Produkt einer Matrix und eines Vektors geschrieben werden, was folgendes ergibt:

$$Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & -S_C C_A D_y - S_C S_A D_z + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -C_C C_A D_y - C_C S_A D_z + C_C D_y \\ 0 & -S_A & C_A & S_A D_y - C_A D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (26)$$

was die Vorwärtstransformation der Kinematik darstellt.

Inverse Transformation

Wir können P aus Gleichung (25) als "P = $({}^{Q}A_P)^{-1}$ * Q" lösen, indem wir wie zuvor (15) verwenden. Wir erhalten somit:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & 0 \\ S_C C_A & C_C C_A & -S_A & -C_A D_y + S_A D_z + D_y \\ S_C S_A & C_S S_A & C_A & -S_A D_y - C_A D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (27)$$

Die gewünschten Gleichungen für die *inverse Transformation* der Kinematik können somit wie folgt geschrieben werden:

$$\begin{aligned} P_x &= C_c Q_x - S_C Q_y \\ P_y &= S_C C_A Q_x + C_C C_A Q_y - S_A Q_z - C_A D_y + S_A D_z + D_y \\ P_z &= S_C S_A Q_x + C_S S_A Q_y + C_A Q_z - S_A D_y - C_A D_z + D_z \end{aligned} \qquad (28)$$

Transformationen für eine xyzbc-trt-Maschine mit Drehachsenverschiebungen

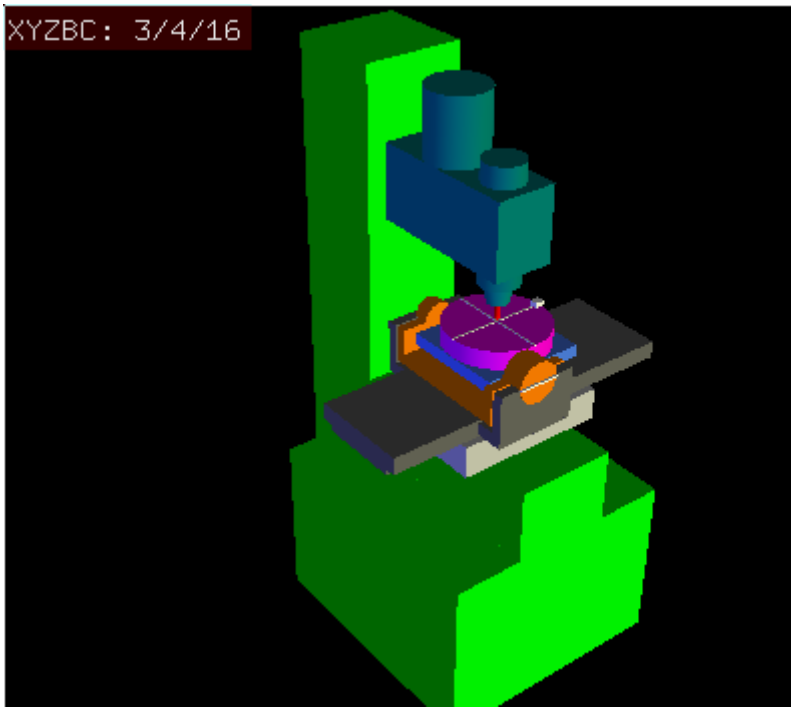


Figure 144. Vismach-Modell von xyzbc-trt mit Rotationsachsenversatz (negativ)

Wir haben es hier wieder mit einer erweiterten Konfiguration zu tun, bei der sich die Kippachse (um die y-Achse) und die Drehachse nicht in einem Punkt schneiden, sondern einen Versatz D_x haben. Außerdem gibt es zwischen den beiden Koordinatensystemen O_{ws} und O_{wp} aus Abb. 2 einen z-Versatz, der D_z genannt wird. Ein Vismach-Modell ist in Abb. 7 dargestellt (negative Versätze in diesem Beispiel), und die positiven Versätze sind in Abb. 8 dargestellt.

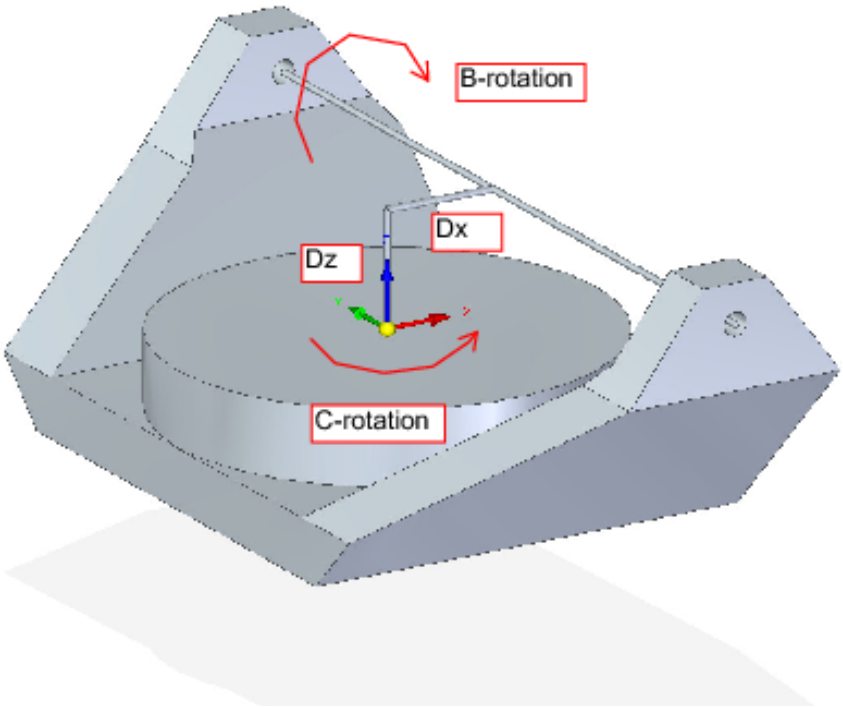


Figure 145. Kipp-/Drehkonfiguration des Tisches xyzbc-trt, mit Achsenversatz

Vorwärts-Transformation

Die Transformation kann durch die sequentielle Multiplikation der Matrizen definiert werden:

$$ {}^{w}A_t = {}^{w}A_O \cdot {}^{O}A_B \cdot {}^{B}A_P \cdot {}^{P}A_t \qquad (29) $$

wobei die Matrizen wie folgt aufgebaut sind:

$$ {}^{w}A_O = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_B = \begin{bmatrix} 1 & 0 & 0 & D_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (30) $$

$$ {}^{B}A_P = \begin{bmatrix} C_B & 0 & -S_B & 0 \\ 0 & 1 & 0 & 0 \\ S_B & 0 & C_B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x - D_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (31) $$

In diesen Gleichungen definieren D_x, D_z die Verschiebungen des Drehpunkts der Drehachsen B relativ zum Ursprung des Werkstückkoordinatensystems. Außerdem sind P_x, P_y, P_z die relativen Abstände des Drehpunkts zur Position der Schneidenspitze, die auch als "Gelenkkoordinaten" des Drehpunkts bezeichnet werden können. Der Drehpunkt liegt auf der B-Drehachse.

Bei Multiplikation gemäß (29) erhalten wir:

$$ {}^{w}A_t = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B & C_C & S_C S_B & -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B & 0 & C_B & S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (32) $$

Wir können nun die dritte Spalte dieser Matrix mit unserem gegebenen Werkzeugorientierungsvektor K gleichsetzen, d. h.:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} -C_C S_B \\ S_C S_B \\ C_B \\ 0 \end{bmatrix} \qquad (33) $$

Aus diesen Gleichungen lassen sich die Drehwinkel $theta_B$, $theta_C$ ermitteln. Aus der dritten Zeile finden wir:

$$ \theta_B = cos^{-1}(K_z) \qquad (0 < \theta_B < \pi) \qquad (34) $$

und durch Division der zweiten Zeile durch die erste Zeile ergibt sich:

$$ \theta_C = tan2^{-1}(K_y, K_x) \qquad (-\pi < \theta_C < \pi) \qquad (35) $$

Diese Beziehungen werden normalerweise im CAM-Postprozessor verwendet, um die Vektoren der Werkzeugausrichtung in Drehwinkel umzuwandeln.

Wenn wir die letzte Spalte von (32) mit dem Werkzeugpositionsvektor Q gleichsetzen, können wir schreiben:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (36) $$

Der Vektor auf der rechten Seite kann auch als das Produkt einer Matrix und eines Vektors geschrieben werden, was folgendes ergibt:

$$Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & -C_c C_B D_x + C_C S_B D_z + C_C D_x \\ -S_C C_B & C_C & S_C S_B & S_C C_B D_x - S_C S_B D_z - S_C D_x \\ S_B & 0 & C_B & -S_B D_x - C_B D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (37)$$

was die Vorwärtstransformation der Kinematik darstellt.

Inverse Transformation

Wir können P aus Gleichung (37) als "P = $({}^{Q}A_{P})^{-1}$ * Q" lösen.

Mit dem gleichen Ansatz wie zuvor, erhalten wir:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & -S_C C_B & S_B & -C_B D_x - S_B D_z + D_x \\ S_C & C_C & 0 & 0 \\ -C_C S_B & S_C S_B & C_B & S_B D_x - C_B D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (38)$$

Die gewünschten Gleichungen für die *inverse Transformation* der Kinematik können somit wie folgt geschrieben werden:

$$\begin{aligned} P_x &= C_c C_B Q_x - S_C C_B Q_y + S_B Q_z - C_B D_x - S_B D_z + D_x \\ P_y &= S_C Q_x + C_C Q_y \\ P_z &= -C_C S_B Q_x + S_C S_B Q_y + C_B Q_z + S_B D_x - C_B D_z + D_z \end{aligned} \qquad (39)$$

9.4.6. Beispiele für Dreh-/Kipptische

LinuxCNC enthält Kinematik-Module für die "xyzac-trt" und "xyzbc-trt" Topologien in der Mathematik oben beschrieben. Für interessierte Benutzer ist der Quellcode im Git-Baum im Verzeichnis "src/emc/kinematics/" verfügbar.

Beispielkonfigurationen für xyzac-trt und xyzbc-trt befinden sich im Verzeichnis Beispielkonfigurationen (*configs/sim/axis/vismach/5axis/table-rotary-tilting/*).

Die Beispielkonfigurationen enthalten die erforderlichen INI-Dateien und ein Beispiel-Unterverzeichnis mit G-Code (NGC) Dateien. Diese sim Konfigurationen rufen ein realistisches 3-dimensionales Modell mit LinuxCNC vismach.

Vismach Simulationsmodelle

Vismach ist eine Bibliothek von Python-Routinen zur Darstellung einer dynamischen Simulation einer CNC-Maschine auf dem PC-Bildschirm. Das Python-Skript für eine bestimmte Maschine wird in HAL geladen und die Daten werden über HAL-Pin-Verbindungen übergeben. Das Nicht-Echtzeit-Vismach-Modell wird durch einen HAL-Befehl wie folgt geladen:

```
loadusr -W xyzac-trt-gui
```

und Verbindungen werden mit HAL-Befehlen hergestellt wie:

```
net  :table-x   joint.0.pos-fb xyzac-trt-gui.table-x
net  :saddle-y  joint.1.pos-fb xyzac-trt-gui.saddle-y
...
```

Einzelheiten zu den HAL-Verbindungen, die für das Vismach-Modell verwendet werden, finden Sie in den INI-Dateien für die Simulation.

Werkzeuglängenkompensation

Um Werkzeuge aus einer Werkzeugtabelle sequentiell mit Werkzeuglängenkompensation automatisch angewendet zu verwenden, ist ein weiterer Z-Offset erforderlich. Für ein Werkzeug, das länger ist als die "Master"-Werkzeug, das typischerweise eine Werkzeuglänge von Null zugewiesen wurde, hat LinuxCNC eine Variable namens "motion.tooloffset.z". Wenn diese Variable auf die kinematische Komponente (und vismach Python-Skript) übergeben wird, dann kann die notwendige zusätzliche Z-Offset für ein neues Werkzeug berücksichtigt werden, indem Sie die Komponente Anweisung, zum Beispiel:

$$D_z = D_z + \text{tool-offset}$$

Die erforderliche HAL-Verbindung (für xyzac-trt) ist:

```
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
```

wo:

```
:tool-offset ---------------- Signalname
motion.tooloffset.z --------- Ausgang HAL-Pin von LinuxCNC Bewegungsmodul
xyzac-trt-kins.tool-offset -- Eingang HAL-Pin zu xyzac-trt-kins
```

9.4.7. Kundenspezifische Kinematik-Komponenten

LinuxCNC implementiert Kinematik mit einer HAL-Komponente, die beim Starten von LinuxCNC geladen wird. Die häufigste Kinematik-Modul, *trivkins*, implementiert Identität (trivial) Kinematik, wo es eine eins-zu-eins-Korrespondenz zwischen einer Achse Koordinate Buchstaben und einem Motor Gelenk. Zusätzliche Kinematik-Module für komplexere Systeme (einschließlich "xyzac-trt" und "xyzbc-trt" oben beschrieben) sind verfügbar.

Kurze Beschreibungen der verfügbaren Kinematikmodule finden Sie in der kins-Manpage (**\$ man kins**).

Die Kinematik-Module von LinuxCNC vorgesehen sind in der Regel in der C-Sprache geschrieben. Durch die Verwendung einer Standardstruktur wird die Erstellung eines benutzerdefinierten Kinematik-Moduls erleichter durch das Kopieren einer vorhandenen Quelldatei in eine Benutzerdatei mit einem neuen Namen, ändern Sie den und dann installieren.

Die Installation erfolgt mit halcompile:

```
sudo halcompile --install kinsname.c
```

wobei "kinsname" der Name ist, den Sie Ihrer Komponente geben. Das sudo-Präfix ist für die Installation erforderlich und Sie werden nach Ihrem root-Passwort gefragt. Weitere Informationen finden Sie in der Manpage von halcompile (**\$ man halcompile**)

Sobald es kompiliert und installiert ist, können Sie es in der Konfiguration Ihres Rechners referenzieren.

Dies geschieht in der INI-Datei Ihres Konfigurationsverzeichnisses. Zum Beispiel die allgemeine INI Spezifikation:

```
[KINS]
KINEMATICS = trivkins
```

wird ersetzt durch

```
[KINS]
KINEMATICS = kinsname
```

wobei "kinsname" der Name Ihres kins-Programms ist. Zusätzliche HAL-Pins können vom Modul für variable Konfigurationselemente wie D_x, D_y, D_z, Werkzeug-Offset, die im xyzac-trt-Kinematikmodul verwendet werden, erstellt werden. Diese Pins können mit einem Signal zur dynamischen Steuerung verbunden werden oder einmalig mit HAL-Verbindungen wie:

```
# Offset-Parameter einstellen
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
setp xyzac-trt-kins.y-versatz 0
setp xyzac-trt-kins.z-versatz 20
```

9.4.8. Abbildungen

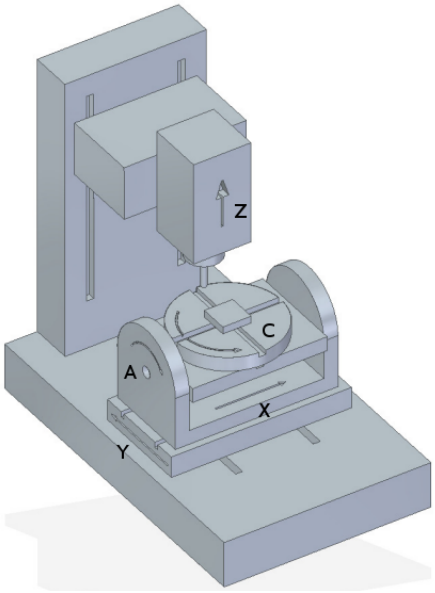


Figure 146. Kipp-/Drehkonfiguration des Tisches

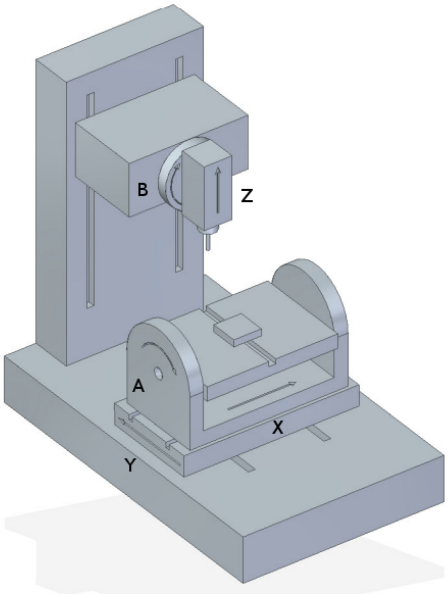


Figure 147. Spindel-/Tischkippkonfiguration

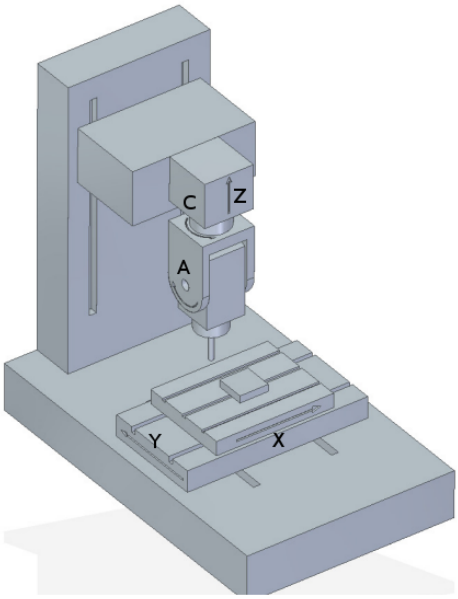


Figure 148. Kipp-/Drehkonfiguration der Spindel

9.4.9. VERWEISE

1. AXIS MACHINE TOOLS: Kinematik und Vismach-Implementierung in LinuxCNC, RJ du Preez, SA-CNC-CLUB, 7. April 2016.
2. A Postprocessor Based on the Kinematics Model for General Five-Axis machine Tools: C-H She, R-S Lee, J Manufacturing Processes, V2 N2, 2000.
3. NC Post-processor for 5-axis milling of table-rotating/tilting type: YH Jung, DW Lee, JS Kim, HS Mok, J Materials Processing Technology,130-131 (2002) 641-646.
4. 3D 6-DOF Serial Arm Robot Kinematics, RJ du Preez, SA-CNC-CLUB, Dec. 5, 2013.
5. Design of a generic five-axis postprocessor based on generalized kinematics model of machine tool: C-H She, C-C Chang, Int. J Machine Tools & Manufacture, 47 (2007) 537-545.

9.5. Schaltbare Kinematik (switchkins)

9.5.1. Einführung

Eine Reihe von Kinematikmodulen unterstützt die Umschaltung von Kinematikberechnungen. Diese Module unterstützen eine Standard-Kinematikmethode (Typ0), eine zweite eingebaute Methode (Typ1) und (optional) eine vom Benutzer bereitgestellte Kinematikmethode (Typ2). Für die Typ1-Methode wird in der Regel die Identitätskinematik verwendet.

Die Switchkins-Funktionalität kann für Maschinen verwendet werden, bei denen eine Steuerung der Gelenke nach der Referenzfahrt während des Einrichtens erforderlich ist oder um Bewegungen in der Nähe von Singularitäten aus dem G-Code zu vermeiden. Solche Maschinen verwenden für die meisten Vorgänge spezifische Kinematikberechnungen, können aber für die Steuerung einzelner Gelenke nach der Referenzfahrt auf Identitätskinematik umgestellt werden.

The kinematics type is selected with *G12.1 P-* and *G13.1*, from a G-code program or by interactive MDI commands. Buttons on a virtual panel (PyVCP, GladeVCP, etc.) or on hardware controls select a kinematics type through the halui provisions for activating MDI commands.

Changing the kinematics type requires the interpreter and motion parts of LinuxCNC to be **synchronized**, which *G12.1* and *G13.1* do themselves.

A deprecated HAL pin, *motion.switchkins-type*, selects a kinematics type as well. It is described under Usage below, because existing configurations use it.

9.5.2. Schaltbare Kinematik-Module

Die folgenden Kinematikmodule unterstützen umschaltbare Kinematiken:

1. **xyzac-trt-kins** (type0:xyzac-trt-kins type1:identity)
2. **xyzbc-trt-kins** (type0:xyzbc-trt-kins type1:identity)
3. **genhexkins** (type0:genhexkins type1:identity)
4. **genserkins** (type0:genserkins type1:identity) (puma560 example)
5. **pumakins** (type0:pumakins type1:identity)
6. **three21kins** (type0:three21kins type1:identity)
7. **scarakins** (type0:scarakins type1:identity)
8. **5axiskins** (type0:5axiskins type1:identity) (bridgemill)
9. **millturn** (type0:identity type1:turn)
10. **xyzab_tdr_kins** (type0:identity type1:tcp)
11. **xyzacb_trsrn** (type0:identity type1:tcp type2:tool)
12. **xyzbca_trsrn** (type0:identity type1:tcp type2:tool)

Every module listed above uses its own kinematics for type0 and identity kinematics for type1. Each accepts the module string parameter *sparm* to swap the two, so that the machine starts in identity

kinematics and the module kinematics are selected on demand:

```
[KINS]
KINEMATICS = xyzac-trt-kins sparm=identityfirst
# ...
```

Starting in identity kinematics leaves a way out of poses the module kinematics cannot solve. A module kinematics failure (near a singularity, for instance) reports an error and disables the machine; type0 is then reached without running the failing kinematics again.

NOTE The *sparm* setting exchanges type0 and type1, so any G-code or HAL logic that selects a kinematics type by number must match.

Kinematics that solve the forward direction iteratively, genhexkins among them, need a pose to start from. While another type is running they take the estimate the caller supplies, which motion seeds from the *[TRAJ]HOME* world home, so they are ready to be switched to.

Identitätsbrief-Zuweisungen

Bei Verwendung eines **identischen** Kinematiktyps kann der Modulparameter *Koordinaten* verwendet werden, um den Gelenken Buchstaben in beliebiger Reihenfolge aus der Menge der zulässigen Koordinatenbuchstaben zuzuweisen. Beispiele:

```
[KINS]
JOINTS = 6

# konventionelle Identitätsanordnung: joint0==x, joint1==y, ...
KINEMATICS = genhexkins coordinates=xyzabc

# custom identity ordering: joint0==c, joint1==b, ...
KINEMATICS = genhexkins coordinates=cbazyx
```

NOTE Wenn der Parameter coordinates= weggelassen wird, lauten die Standard-Zuordnungen der Gelenkbuchstaben joint0==x,joint1=y,....

Die Gelenkzuweisungen für **Identitäts**-Kinematiken bei Verwendung des Koordinatenparameters sind identisch mit denen für das Modul trivkins. Die Duplizierung von Achsenbuchstaben zur Zuweisung mehrerer Gelenke für einen Koordinatenbuchstaben ist jedoch im Allgemeinen nicht für serielle oder parallele Kinematiken (wie genserkins, pumakins, genhexkins usw.) geeignet, bei denen es keine einfache Beziehung zwischen Gelenken und Koordinaten gibt.

Die Duplizierung von Achskoordinatenbuchstaben wird in den Kinematikmodulen xyzac-trt-kins, xyzbc-trt-kins und 5axiskins (bridgemill) unterstützt. Typische Anwendungen für doppelte Koordinaten sind Gantry-Maschinen, bei denen zwei Motoren (Gelenke) für die Querachse verwendet werden.

Rückwärtskompatibilität

Schaltbare Kinematiken werden mit motion.switchkins-type==0 initialisiert und implementieren ihre

gleichnamige Kinematikmethode. Wenn der motion.switchkins-type-Pin nicht angeschlossen ist - wie in Legacy-Konfigurationen - ist nur der Standard-Kinematik-Typ verfügbar.

9.5.3. HAL-Pins

Die Umschaltung der Kinematik wird durch den Motion-Modul-Eingang HAL pin **motion.switchkins-type** gesteuert. Der Fließkommawert des Pins wird in eine Ganzzahl umgewandelt und zur Auswahl eines der angebotenen Kinematik-Typen verwendet. Der Startwert Null wählt den Standard-Kinematiktyp Typ0.

NOTE	Der Eingangspin motion.switchkins-type ist ein Fließkomma-Eingangspin, um den Anschluss an die Ausgangspins des Motion-Moduls wie motion.analog-out-0n zu erleichtern, die von Standard M-Codes (typischerweise M68EnL0) gesteuert werden können.

Es sind Ausgangs-HAL-Pins vorgesehen, um GUIs über den aktuellen Kinematik-Typ zu informieren. Diese Pins können auch mit digitalen Eingängen verbunden werden, die von G-Code-Programmen gelesen werden, um das Programmverhalten entsprechend dem aktiven Kinematik-Typ zu aktivieren oder zu deaktivieren.

HAL Pin Zusammenfassung

1. **motion.switchkins-type** Input (real)
2. **motion.kins-type** Output (real)
3. **kinstype.is-0** Output (bool)
4. **kinstype.is-1** Output (bool)
5. **kinstype.is-2** Output (bool)

A module providing more than three kinematics types has one *kinstype.is-N* pin per type.

9.5.4. Anwendung

HAL-Verbindungen

G12.1 and *G13.1* ask motion for a kinstype directly and need no HAL connection at all.

A kinstype can also be selected by writing the pin **motion.switchkins-type**, which is sourced from an analog output pin like motion.analog-out-03 so that it can be set by M68 commands:

```
net :kinstype-select <= motion.analog-out-03
net :kinstype-select => motion.switchkins-type
```

WARNING	Selecting the kinstype from HAL is deprecated and motion says so, once, the first time the pin is used to change it. The interpreter does not see the pin, so a program is read, its limits checked and its path looked ahead in whatever kinematics the

interpreter last knew about, which is not necessarily the one that will run it. Use *G12.1* and *G13.1*. The pin is in a grace period: it keeps working for now, but is meant to be removed in the future.

G-code commands

G12.1 P- selects a kinstype and *G13.1* cancels back to identity kinematics. Which kinstype is identity is declared by the module (see Code Notes), not fixed to a number:

```
...
G12.1 P1  ;select kinstype 1
...
...       ;user G-code
...
G13.1     ;back to identity kinematics
...
```

These codes ask motion for the kinstype directly and synchronize task and motion themselves, so no HAL connection and no separate sync command are needed. The G-code words and the **motion.switchkins-type** pin are both acted on when they change, so whichever asked most recently is the one in force. **motion.kins-type** reports what is currently selected.

The pin is deprecated, see the warning under HAL Connections.

The kinstype in force is readable in G-code as *#<_kins_type>*, which lets a subroutine restore whatever its caller had selected:

```
#<saved> = #<_kins_type>
G12.1 P2
( ... )
G12.1 P#<saved>
```

Selection is not cancelled by the end of a program or by an abort, so that the kinstype continues to match the position readout. A program that should leave the machine in identity kinematics ends with *G13.1*.

A module that declares no identity kinstype refuses *G13.1* with an error and can still be driven by number with *G12.1*; see Code Notes for how a module declares its types.

For tool length work there are spellings that name the kinematics by what it is rather than by number: *G43.4* applies the tool length offset and switches to the kinstype the module declares its working transform, and the *G49* that cancels it switches back to identity. See the G-code documentation for *G43.4* and *G49*, and for *G12.1* and *G13.1*, for the full description.

M-code commands

WARNING This is the deprecated route described under HAL Connections above. It is documented because existing configurations use it. New ones should use *G12.1* and *G13.1*.

Writing **motion.switchkins-type** through an analog output pin needs the HAL connection shown above. Kinstype selection is then managed using G-code sequences like:

```
...
M68 E3 Q1 ;analog-out-03 aktualisieren, um Kinstype 1 auszuwählen
M66 E0 L0 ;Sync Interp-Bewegung
...
... ;Benutzer G-Code
...
M68 E3 Q0 ;analog-out-03 aktualisieren, um Kinstype 0 zu wählen
M66 E0 L0 ;Sync Interp-Bewegung
...
```

NOTE Ein M66-Befehl *wait-on-input* aktualisiert die Variable #5399. Wenn der aktuelle Wert dieser Variablen für spätere Zwecke benötigt wird, sollte er vor dem Aufruf von M66 in eine zusätzliche Variable kopiert werden.

Diese G-Code-Befehlssequenzen werden in der Regel in G-Code-Unterprogrammen als remapped M-codes oder mit herkömmlichen M-code-Skripten implementiert.

Vorgeschlagene Codes (wie in den Sim-Konfigurationen verwendet) sind:

Herkömmliche Benutzer-M-Codes:

1. M128 Kintyp 0 auswählen (Standardkinematik beim Start)
2. M129 Kintyp 1 auswählen (typischerweise Identitätskinematik)
3. M130 Kinstype 2 auswählen (benutzerdefinierte Kinematik)

Neu zugeordnete M-Codes:

1. M428 Kintyp 0 auswählen (Standardkinematik beim Start)
2. M429 Kinstype 1 auswählen (typischerweise Identitätskinematik)
3. M430 Kinstype 2 auswählen (benutzerdefinierte Kinematik)

NOTE Herkömmliche Benutzer-M-Codes (im Bereich M100-M199) gehören zur Modalgruppe 10. Neu zugeordnete M-Codes (im Bereich M200 bis M999) können eine Modalgruppe angeben. Weitere Informationen finden Sie in der Remap-Dokumentation.

INI-Datei Limit Einstellungen

LinuxCNC Bahnplanung verwendet Grenzen für die Position (min, max), Geschwindigkeit und Beschleunigung für jede anwendbare Koordinaten-Buchstaben in der Konfiguration INI-Datei angegeben. Beispiel für den Buchstaben L (im Satz *XYZABCUVW*):

```
[AXIS_L]
MIN_LIMIT =
MAX_LIMIT =
MAX_VELOCITY =
```

```
MIN_ACCELERATION =
```

Die angegebenen INI-Datei-Grenzwerte gelten für die Standardkinematik vom Typ 0, die beim Start aktiviert wird. Beim Umschalten auf eine andere Kinematik sind diese Grenzen möglicherweise **nicht** anwendbar. Da jedoch beim Umschalten der Kinematik eine Synchronisierung zwischen Interpreter und Bewegung erforderlich ist, können INI-HAL-Pins verwendet werden, um Grenzwerte für einen anstehenden Kinematik-Typ festzulegen.

NOTE INI-HAL-Pins werden während eines G-Code-Programms normalerweise nicht erkannt, es sei denn, es wird einw Synchronisations (der sogenannte Queue-Buster, engl. für "wartende Befehle Zerstörer") ausgegeben. Weitere Informationen hierzu finden Sie in der milltask-Manpage ($ man milltask).

Die für eine gemeinsame Nummer (*N*) relevanten INI-HAL-Pins sind:

```
ini.N.min_limit
ini.N.max_limit
ini.N.max_acceleration
ini.N.max_velocity
```

Die für eine Achsenkoordinate (*L*) relevanten INI-HAL Pins sind:

```
ini.L.min_limit
ini.L.max_limit
ini.L.max_velocity
ini.L.max_acceleration
```

NOTE Im Allgemeinen gibt es keine festen Zuordnungen zwischen Gelenknummern und Achsenkoordinatenbuchstaben. Für einige Kinematikmodule, insbesondere solche, die Identitätskinematik implementieren (trivkins), kann es spezifische Zuordnungen geben. Weitere Informationen finden Sie in der kins man page ($ man kins).

Ein vom Benutzer bereitgestellter M-code kann eine oder alle der Achsenkoordinaten Grenzen vor der Änderung der motion.switchkins-type Pin und die Synchronisierung der Interpreter und Motion-Teile von LinuxCNC ändern. Als Beispiel kann ein Bash-Skript, das halcmd aufruft, "hardcoded" werden, um eine beliebige Anzahl von HAL-Pins zu setzen:

```
#!/bin/bash
halcmd -f <<EOF
setp ini.x.min_limit -100
setp ini.x.max_limit  100
# ... repeat for other limit parameters
EOF
```

Skripte wie dieses können als Benutzer-M-Code aufgerufen und **vor** dem Kinstype-Switching-Mcode verwendet werden, der den `motion.switchkins-type` HAL Pin aktualisiert und einen interp-motion-Sync erzwingt. Normalerweise würden für jeden kinstype (0,1,2) separate Skripte verwendet werden.

Wenn Identitätskinematiken als Mittel zur Steuerung einzelner Gelenke vorgesehen sind, kann es sinnvoll sein, die in der System-INI-Datei angegebenen Grenzwerte festzulegen oder wiederherzustellen. Ein Beispiel: Ein Roboter startet nach der Referenzfahrt mit einer komplexen (nicht identischen) Kinematik (Typ 0). Das System ist so konfiguriert, dass es auf eine Identitätskinematik (Typ1) umgeschaltet werden kann, um einzelne Gelenke mit den herkömmlichen Buchstaben aus dem Satz *XYZABCUVW* zu manipulieren. Die Einstellungen in der INI-Datei ([AXIS_L]) sind beim Betrieb mit Identitätskinematik (Typ1) **nicht** anwendbar. Um diesem Anwendungsfall gerecht zu werden, können die Benutzer-M-code-Skripte wie folgt gestaltet werden:

M129 (Umschalten auf Identitätstyp1)

1. INI-Datei lesen und auswerten ("parsen")
2. hal: setzt die INI-HAL Grenzstifte für jeden Achsenbuchstaben ([AXIS_L]) entsprechend der *identitätsbezogenen* Gelenknummer INI-Datei ([JOINT_N])
3. HAL: `setp motion.switchkins-type 1`
4. MDI: Ausführen eines Synchronisations-G-Codes (M66E0L0)

M128 (Wiederherstellung der Standardkinematik des Roboters, Typ 0)

1. INI-Datei lesen und auswerten ("parsen")
2. HAl: Setzen der INI-HAL Limit Pins für jeden Achsenbuchstaben ([AXIS_L]) entsprechend der entsprechenden INI-Datei Einstellung ([AXIS_L])
3. HAL: `setp motion.switchkins-type 0`
4. MDI: Ausführen eines Synchronisations-G-Codes (M66E0L0)

NOTE Die Vismach-Simulationskonfigurationen für einen Puma-Roboter demonstrieren die M-Code-Skripte (M128, M129, M130) für diesen Beispielanwendungsfall.

Koordinatensystem-Offsets Berücksichtungen

Wie die Limit Einstellungen in der INI-Datei gelten auch die Koordinatensystem-Offsets (G92, G10L2, G10L20, G43 usw.) im Allgemeinen nur für die Standard-Startkinematik vom Typ 0. Beim Wechsel des Kinematik-Typs kann es **wichtig** sein, entweder alle Offsets vor dem Wechsel zurückzusetzen oder die Offsets entsprechend den systemspezifischen Anforderungen zu aktualisieren.

Externe Offsets Berücksichtungen

Externe Offsets (gesetzt auf eine Achse (*L*) über axis.*L*.eoffset-request) bleiben bei Kinematikwechseln erhalten. Wenn ein Offset auf einer Achse vor dem Wechsel aktiv ist (sichtbar in axis.*L*.eoffset), behält der Trajektorienplaner dieses Offset nach dem Wechsel bei, ähnlich wie er die befohlene Position aus einem G-Code beibehält. Dies gewährleistet ein konsistentes Maschinenverhalten unabhängig von der aktiven Kinematik.

Wenn das Beibehalten des Offsets aufgrund von Achsgrenzenänderungen oder anderen Faktoren problematisch sein könnte, stellen Sie sicher, dass das eoffset vor dem Kinematikwechsel gelöscht und gegebenenfalls deaktiviert wird.

9.5.5. Simulationskonfigurationen

Simulationskonfigurationen (die keine Hardware erfordern) werden mit illustrativen Vismach-Anzeigen in Unterverzeichnissen von configs/sim/axis/vismach/ bereitgestellt.

1. 5axis/table-rotary-tilting/xyzac-trt.ini (xyzac-trt-kins)
2. 5axis/table-rotary-tilting/xyzbc-trt.ini (xyzac-trt-kins)
3. 5axis/bridgemill/5axis.ini (5axiskins)
4. scara/scara.ini (scarakins)
5. puma/puma560.ini (genserkins)
6. puma/puma.ini (pumakins)
7. hexapod-sim/hexapod.ini (genhexkins)
8. millturn/millturn.ini (millturn)
9. 5axis/table-dual-rotary/xyzab-tdr.ini (xyzab_tdr_kins)
10. 5axis/table-rotary_spindle-rotary-nutating/xyzacb-trsrn_twp/xyzacb-trsrn.ini (xyzacb_trsrn)
11. 5axis/table-rotary_spindle-rotary-nutating/xyzbca-trsrn_twp/xyzbca-trsrn.ini (xyzbca_trsrn)

9.5.6. Kinematische Bestimmungen des Benutzers

There are two ways to supply custom kinematics. Adding a kinstype to a module that is already in the tree is the smaller job; building a module of your own gives you every kinstype it provides.

Adding a kinstype to an in-tree module

Benutzerdefinierte Kinematiken können auf Run-In-Place ("RIP") Builds kodiert und getestet werden. Eine Vorlagendatei src/emc/kinematics/userkfuncs.c ist in der Distribution enthalten. Diese Datei kann in ein Benutzerverzeichnis kopiert/umbenannt und bearbeitet werden, um benutzerdefinierte Kinematik mit kinstype==2 bereitzustellen.

Die benutzerdefinierte Kinematikdatei kann bei rt-preempt-Implementierungen aus den Out-of-Tree-Quellen kompiliert werden oder bei rtai-Systemen durch Ersetzen der In-Tree-Vorlagendatei (src/emc/kinematics/userkfuncs.c).

Preempt-rt make Beispiel:

```
$ userkfuncs=/home/myname/kins/mykins.c make && sudo make setuid
```

Building a switchkins module of your own

A complete kinematics module can be built out-of-tree with halcompile using the same switchkins implementation the in-tree modules use, so it gets the kinematics switching, the *kinstype.is-N* pins, the *coordinates*= identity mapping and the G-code and HAL controls without reimplementing any of them.

The implementation is the switchkins_core module. It provides no kinstypes of its own; it exports

switchkinsRegister(), switchkinsInit() and the rest of the switchkins.h interface, and the identity kinematics and helpers kinematics.h declares. A module of your own includes switchkins.h, registers each of its kinstypes and calls switchkinsInit() from EXTRA_SETUP(), which halcompile runs after hal_init() and before hal_ready(). See Code Notes for both calls.

The template is src/hal/components/switchkinscomp.comp. Copy and rename it (both the file and the component name) and replace the example kinstype with the real kinematics. Start from the template, not from an in-tree component such as millturn.comp: those link the switchkins objects into the module through hal/components/Submakefile, which a halcompile build outside the tree does not do.

```
$ halcompile --install user_switchkins.comp
```

```
[KINS]
KINEMATICS = user_switchkins
JOINTS = 3
```

The HAL file loads switchkins_core ahead of the module:

```
loadrt switchkins_core
loadrt [KINS]KINEMATICS
```

A module loaded without switchkins_core ahead of it fails to load, and the error names the first switchkins or kinematics function it could not find:

```
user_switchkins: dlopen: .../rtlib/user_switchkins.so: undefined symbol: ...
```

9.5.7. Warnungen

Unerwartetes Verhalten kann auftreten, wenn ein G-Code-Programm versehentlich mit einem inkompatiblen Kinematik-Typ gestartet wird. Unerwünschtes Verhalten kann in G-Code-Programmen umgangen werden, indem:

1. Anschluss geeigneter kinstype.is.N HAL-Pins an digitale Eingangspins (wie motion.digital-in-0m).
2. Auslesen des digitalen Eingangspins (M66 E0 Pm) beim Start des G-Code-Programms
3. Abbruch (M2) des G-Code-Programms mit einer Meldung (DEBUG, problem_message), wenn der Kintyp nicht geeignet ist.

Bei der interaktiven Verwendung von Jogging-Einrichtungen oder MDI-Befehlen ist Vorsicht geboten. Leitfäden sollten Anzeigen enthalten, die den aktuellen Kinematik-Typ anzeigen.

NOTE Die Umstellung auf eine andere Kinematik kann erhebliche betriebliche Veränderungen mit sich bringen, die eine sorgfältige Planung, Prüfung und Schulung für den Einsatz erfordern. Die Verwaltung von Koordinatenversatz, Werkzeugkompensation und INI-Datei Limits kann komplizierte und nicht standardisierte Betriebsprotokolle erfordern.

9.5.8. Code Anmerkungen

Kinematic modules providing switchkins functionality are linked to the switchkins.o object (switchkins.c). It provides kinematicsForward(), kinematicsInverse(), kinematicsSwitch() and the rest of the kinematics interface, dispatching each call to the kinstype currently selected, and it creates the HAL pins common to all switchkins modules. It does not provide the module *main* program, so a module can get that from wherever suits it.

A kinstype is supplied by calling switchkinsRegister(), once per kinstype:

```
int switchkinsRegister(int ktype, KS kset, KF kfwd, KI kinv);
```

ktype runs from 0 to SWITCHKINS_MAX_TYPES-1 (defined in kinematics.h). A kinstype has to come from one route or the other, so registering one that switchkinsSetup() has already filled in is an error, and so is leaving a gap below the highest kinstype provided. Either mistake fails the module load and says which kinstype is at fault.

Each kinstype gets its own *kinstype.is-N* pin, so a module providing the usual three keeps the pin names it always had.

A module also declares what each kinstype IS, with flags from kinematics.h:

1. **KINSTYPE_IDENTITY** no transform: the joints are the world
2. **KINSTYPE_PRIMARY** the module's working transform

A kinstype registered with switchkinsRegisterOps() carries its flag in the ops table itself, as the *identity* or *primary* field; a kinstype registered the older way gets it from a call, again from within switchkinsSetup():

```
int switchkinsDeclare(int ktype, int flags);
```

G-code reads these declarations: *G13.1* cancels to the kinstype declared KINSTYPE_IDENTITY, and *G43.4* switches to the kinstype declared KINSTYPE_PRIMARY, whatever their numbers, so a module whose kinematics are not in the conventional order still gets working spellings. At most one kinstype may be declared identity and at most one primary, and declaring a kinstype the module does not provide fails the module load. A module that declares nothing keeps working exactly as before for *G12.1 P-* and *G49*, but *G13.1* and *G43.4* are an error, since the numbers of the identity and primary kinematics are then a guess.

A module that implements kinematicsTypeFlags() itself, without switchkins.c, returns the flags of every kinstype it can switch to, 0 when neither applies, and -1 for any other: *G12.1 P-* refuses a kinstype that returns -1.

When every kinstype is registered, the module calls:

```
int switchkinsInit(const int comp_id, kparms* kp, const char* coordinates);
```

which checks the supplied parameters, creates the HAL pins, selects kinstype 0, and then invokes the

setup routine registered for each kinstype. The caller owns the HAL component: it does hal_init() before switchkinsInit() and hal_ready() after it.

Each kinstype setup routine can (optionally) create HAL pins and set them to default values. A setup routine is called once per kinstype it is registered for, so a routine used for two kinstypes must not create the same pin twice.

Module main program

A module written as a plain C file links switchkins_main.o (switchkins_main.c) for its rtapi_app_main(). That *main* program reads the (optional) module command-line parameters (coordinates, sparm) and passes them to the module-provided function switchkinsSetup():

```
int switchkinsSetup(kparms* kp,
                    KS* kset0, KS* kset1, KS* kset2,
                    KF* kfwd0, KF* kfwd1, KF* kfwd2,
                    KI* kinv0, KI* kinv1, KI* kinv2);
```

which identifies the setup, forward and inverse routines for kinstypes 0,1,2 and sets a number of configuration settings. Those three are registered for the module, so it can supply further kinstypes by calling switchkinsRegister() itself, and registering one that switchkinsSetup() has already filled in is the same error as any other duplicate.

A module written as a halcompile component gets rtapi_app_main() from halcompile instead. It registers its kinstypes and calls switchkinsInit() from its EXTRA_SETUP() routine, which halcompile runs after hal_init() and before hal_ready():

```
EXTRA_SETUP() {
    kparms kp;
    (void)__comp_inst; (void)prefix; (void)extra_arg;

    kp.kinsname = "mykins"; // must agree with the module name
    // ... the other kparms settings
    if (switchkinsRegister(0, identityKinematicsSetup,
                           identityKinematicsForward,
                           identityKinematicsInverse)) { return -1; }
    if (switchkinsRegister(1, mySetup, myForward, myInverse)) { return -1; }
    return switchkinsInit(comp_id, &kp, coordinates);
}
```

Where the switchkins code comes from depends on where the component is built. Out of tree, the switchkins_core module provides it, loaded ahead of the component (see Building a switchkins module of your own). In tree, the component links the objects into itself instead, named in hal/components/Submakefile:

```
millturn-extra-objs := emc/kinematics/switchkins.o emc/kinematics/kins_util.o
```

Outline

The two routes in one switchkinsSetup(), with the kinematics itself left out. Types 0 to 2 are filled in through the pointer arguments as they always were, and a fourth is registered:

```
int switchkinsSetup(kparms* kp,
                    KS* kset0, KS* kset1, KS* kset2,
                    KF* kfwd0, KF* kfwd1, KF* kfwd2,
                    KI* kinv0, KI* kinv1, KI* kinv2
                   )
{
    kp->kinsname             = "mykins"; // must agree with the filename
    kp->halprefix            = "mykins"; // hal pin names
    kp->required_coordinates = "xyzab";
    kp->max_joints           = strlen(kp->required_coordinates);
    // remaining kparms fields

    *kset0 = identityKinematicsSetup;    // kinstype 0 is the startup default
    *kfwd0 = identityKinematicsForward;
    *kinv0 = identityKinematicsInverse;

    *kset1 = myKinematicsSetup;
    *kfwd1 = myKinematicsForward;
    *kinv1 = myKinematicsInverse;

    *kset2 = userkKinematicsSetup;
    *kfwd2 = userkKinematicsForward;
    *kinv2 = userkKinematicsInverse;

    // any further kinstype comes from switchkinsRegister(), and the
    // numbering carries on from the three above with no gaps
    if (switchkinsRegister(3, myOtherKinematicsSetup,
                           myOtherKinematicsForward,
                           myOtherKinematicsInverse)) { return -1; }

    return 0;
} // switchkinsSetup()
```

A module wanting fewer than three kinstypes leaves the unused pointer arguments alone and starts registering at the first free number.

For the surrounding shape, the in-tree switchkinsSetup() routines are in src/emc/kinematics: 5axiskins.c, xyzac-trt-kins.c, genserkins.c, scarakins.c and the others listed at the top of this document. None of them registers a fourth kinstype yet, so the call above has no in-tree example to copy.

9.6. PID-Feineinstellung (engl. tuning)

9.6.1. PID-Regler (engl. PID controller)

Ein Proportional-Integral-Derivativ-Regler (PID-Regler) ist eine gängige Rückkopplungskomponente in

industriellen Steuerungssystemen. [3]

Der Regler vergleicht einen Messwert aus einem Prozess (in der Regel ein industrieller Prozess) mit einem Referenzsollwert. Die Differenz (oder das "Fehlersignal") wird dann verwendet, um einen neuen Wert für einen manipulierbaren Eingang in den Prozess zu berechnen, der den Prozessmesswert wieder auf den gewünschten Sollwert bringt.

Im Gegensatz zu einfacheren Regelalgorithmen kann der PID-Regler die Prozessausgänge auf der Grundlage des Verlaufs und der Änderungsrate des Fehlersignals anpassen, was eine genauere und stabilere Regelung ermöglicht. (Es lässt sich mathematisch nachweisen, dass eine PID-Regelschleife in Fällen, in denen eine einfache proportionale Regelung entweder einen stationären Fehler aufweisen oder den Prozess zum Schwingen bringen würde, eine genaue, stabile Regelung ergibt).

Grundlagen des Regelkreises

Intuitiv versucht die PID-Schleife das zu automatisieren, was ein intelligenter Bediener mit einem Messgerät und einem Regelknopf tun würde. Der Bediener würde ein Messgerät ablesen, das den Ausgangsmesswert eines Prozesses anzeigt, und den Drehknopf verwenden, um den Eingang des Prozesses (die "Aktion") anzupassen, bis sich der Ausgangsmesswert des Prozesses auf dem gewünschten Wert auf dem Messgerät stabilisiert.

In der älteren Steuerungsliteratur wird dieser Einstellvorgang als "Rückstellung" bezeichnet. Die Position der Nadel auf dem Messgerät ist eine "Messung", ein "Prozesswert" oder eine "Prozessvariable". Der gewünschte Wert auf dem Messgerät wird als "Sollwert" bezeichnet (auch "Einstellwert" genannt). Die Differenz zwischen der Nadel des Messgeräts und dem Sollwert ist der "Fehler".

Ein Regelkreis besteht aus drei Teilen:

1. Messung durch einen an den Prozess angeschlossenen Sensor (z. B. Encoder),
2. Entscheidung in einem Steuerungselement,
3. Aktion durch ein Ausgabegerät wie z. B. einen Motor.

Wenn der Regler einen Sensor abliest, subtrahiert er diese Messung vom "Sollwert", um den "Fehler" zu ermitteln. Anhand des Fehlers berechnet er dann eine Korrektur der Eingangsvariablen des Prozesses (die "Aktion"), so dass diese Korrektur den Fehler aus der Ausgangsmessung des Prozesses entfernt.

In einer PID-Schleife wird die Korrektur auf drei Arten aus dem Fehler berechnet: Der aktuelle Fehler wird direkt ausgeglichen (Proportional), die Zeit, in der ein Fehler unkorrigiert geblieben ist (Integral), und der zukünftige Fehler wird aus der Änderungsrate des Fehlers über die Zeit vorweggenommen (Derivativ).

Ein PID-Regler kann zur Regelung jeder messbaren Größe verwendet werden, die durch die Beeinflussung einer anderen Prozessgröße beeinflusst werden kann. Er kann zum Beispiel zur Regelung von Temperatur, Druck, Durchfluss, chemischer Zusammensetzung, Geschwindigkeit oder anderen Variablen eingesetzt werden. Ein Beispiel für einen Prozess außerhalb der Industrie, bei dem eine grobe PID-Regelung zum Einsatz kommt, ist die Geschwindigkeitsregelung von Autos.

Einige Regelsysteme ordnen PID-Regler in Kaskaden oder Netzwerken an. Das heißt, ein "Master"-Regler

erzeugt Signale, die von "Slave"-Reglern verwendet werden. Eine häufige Situation sind Motorsteuerungen: Oft soll der Motor eine geregelte Drehzahl haben, wobei der "Slave"-Regler (oft in einen Frequenzumrichter eingebaut) die Drehzahl direkt auf der Grundlage eines proportionalen Eingangs steuert. Dieser *Slave*-Eingang wird vom Ausgang des *Master*-Reglers gespeist, der auf der Grundlage einer verwandten Variablen regelt.

Theorie

PID ist nach seinen drei korrigierenden Berechnungen benannt, die alle die kontrollierte Menge ergänzen und anpassen. Diese Additionen sind eigentlich "Subtraktionen" von Fehlern, da die Proportionen normalerweise negativ sind:

Proportional

Dazu wird die Regelabweichung mit einer (negativen) Konstante P (für "proportional") multipliziert und zur Regelgröße addiert (und die Regelabweichung davon subtrahiert). P ist nur in dem Bereich gültig, in dem der Ausgang eines Reglers proportional zur Regelabweichung des Systems ist. Ist die Regelabweichung gleich Null, dann ist der Ausgang eines Proportionalreglers gleich Null.

Integral

Um aus der Vergangenheit zu lernen, wird die Abweichung über einen bestimmten Zeitraum integriert (aufaddiert), dann mit einer (negativen) Konstante I multipliziert (ein Mittelwert gebildet) und zur Regelgröße addiert (die Abweichung wird von ihr subtrahiert). I mittelt die gemessene Abweichung, um die durchschnittliche Abweichung des Prozessausgangs vom Sollwert zu ermitteln. Ein einfaches proportionales System schwingt entweder hin und her um den Sollwert, weil es nichts gibt, um die Abweichung zu beseitigen, wenn es über den Sollwert hinausgeht, oder es schwingt und/oder stabilisiert sich bei einem zu niedrigen oder zu hohen Wert. Indem ein negativer Anteil des durchschnittlichen Fehlers zum Prozesseingang addiert (d. h. ein Teil davon abgezogen) wird, verringert sich immer weiter die durchschnittliche Differenz zwischen dem Prozessausgang und dem Sollwert. Daher wird sich der Prozessausgang einer gut abgestimmten PID-Schleife schließlich auf den Sollwert einpendeln.

Ableitung

Für die Zukunft wird die erste Ableitung (die Steigung der Regelabweichung) nach der Zeit berechnet und mit einer anderen (negativen) Konstante D multipliziert und ebenfalls zur Regelgröße addiert (und die Regelabweichung davon abgezogen). Der Ableitungsterm steuert die Reaktion auf eine Änderung im System. Je größer der Ableitungsterm ist, desto schneller reagiert der Regler auf Änderungen im Ausgang des Prozesses.

Technisch gesehen kann eine PID-Schleife als ein Filter charakterisiert werden, der auf ein komplexes System im Frequenzbereich angewendet wird. Dies ist nützlich, um zu berechnen, ob tatsächlich ein stabiler Wert erreicht wird. Werden die Werte falsch gewählt, kann der Eingang des geregelten Prozesses schwanken und der Ausgang des Prozesses bleibt möglicherweise nie auf dem Sollwert.

Schleifenabstimmung (engl. loop tuning)

Das *Tuning* eines Regelkreises ist die Anpassung seiner Regelparameter (Verstärkung/Proportionalbereich, Integralverstärkung/Rückstellung, Ableitungsverstärkung/Rate) an die optimalen Werte für das gewünschte Regelverhalten. Das optimale Verhalten bei einer Prozess- oder

Sollwertänderung hängt von der jeweiligen Anwendung ab. Bei einigen Prozessen darf die Prozessvariable nicht über den Sollwert hinausschießen. Bei anderen Prozessen muss der Energieaufwand für das Erreichen eines neuen Sollwerts minimiert werden. Im Allgemeinen ist eine stabile Reaktion erforderlich, und der Prozess darf bei keiner Kombination von Prozessbedingungen und Sollwerten schwanken.

Die Abstimmung von Regelkreisen wird durch die Reaktionszeit des Prozesses erschwert; es kann Minuten oder Stunden dauern, bis eine Sollwertänderung eine stabile Wirkung zeigt. Einige Prozesse weisen einen gewissen Grad an Nichtlinearität auf, so dass Parameter, die unter Volllastbedingungen gut funktionieren, beim Anfahren des Prozesses im Leerlauf nicht funktionieren. In diesem Abschnitt werden einige herkömmliche manuelle Methoden zur Regelkreisabstimmung beschrieben.

Es gibt mehrere Methoden zur Abstimmung einer PID-Schleife. Die Wahl der Methode hängt weitgehend davon ab, ob die Schleife für die Abstimmung "offline" genommen werden kann oder nicht, sowie von der Reaktionsgeschwindigkeit des Systems. Wenn das System offline geschaltet werden kann, besteht die beste Abstimmungsmethode oft darin, das System einer sprunghaften Änderung des Eingangs zu unterziehen, den Ausgang als Funktion der Zeit zu messen und diese Reaktion zur Bestimmung der Regelparameter zu verwenden.

Einfache Methode

Wenn das System am Netz bleiben muss, besteht eine Abstimmungsmethode darin, zunächst die Werte für I und D auf Null zu setzen. Erhöhen Sie den P-Wert, bis der Ausgang der Schleife schwingt. Dann erhöhen Sie I, bis die Oszillation aufhört. Schließlich erhöhen Sie D, bis die Schleife ihren Sollwert akzeptabel schnell erreicht. Bei einer schnellen PID-Schleifenabstimmung kommt es in der Regel zu einem leichten Überschwingen, um den Sollwert schneller zu erreichen; einige Systeme können jedoch kein Überschwingen akzeptieren.

Parameter	Anstiegszeit (engl. rise time)	Überschwingen	Eingewöhnungszeit (engl. settling time)	Fehler im eingeschwungenen Zustand
P	Verringerung	Erhöhung	Kleine Veränderung	Verringerung
I	Verringerung	Erhöhung	Erhöhung	Eliminieren
D	Kleine Veränderung	Verringerung	Verringerung	Kleine Veränderung

Auswirkungen steigender Parameter

Ziegler-Nichols-Methode Eine weitere Abstimmungsmethode ist formal bekannt als die

Ziegler-Nichols-Methode, eingeführt von John G. Ziegler und Nathaniel B. Nichols 1942 F [4].]. Es beginnt auf die gleiche Weise wie die zuvor beschriebene Methode: Stellen Sie zuerst die I- und D-Verstärkung auf Null und erhöhen Sie dann die P-Verstärkung und setzen Sie die Schleife externen Störungen aus, z. B. indem Sie die Motorachse anklopfen, um sie aus dem Gleichgewicht zu bringen, um die kritische Verstärkung und die Schwingungsperiode zu bestimmen, bis der Ausgang der Schleife zu schwingen beginnt. Notieren Sie die kritische Verstärkung (K_c) und die Schwingungsperiode des Ausgangs (P_c). Passen Sie dann die Steuerelemente P, I und D an, wie in der Tabelle gezeigt:

Steuerungstyp	P	I	D
P	$.5K_c$		
PI	$.45K_c$	$P_c/1.2$	
PID	$.6K_c$	$P_c/2$	$P_c/8$

Letzte Schritte

Nach der Einstellung der Achse überprüfen Sie den folgenden Fehler mit Halscope, um sicherzustellen, dass er den Anforderungen Ihrer Maschine entspricht. Weitere Informationen zu Halscope finden Sie in der HAL-Bedienungsanleitung.

Automatische PID-Abstimmung

Seit LinuxCNC Version 2.9 unterstützt die pid-Komponente die automatische Abstimmung mit der Relay-Methode Fußnote:[Åström, Karl Johan und Hägglund, Tore (1984), *Automation paper Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins*, DOI 10.1016/0005-1098(84)90014-1]. Dies ist ein Ersatz für die jetzt entfernte und veraltete at_pid-Komponente.

Die PID-Komponente verwendet mehrere Konstanten, um den Ausgangswert auf der Grundlage des aktuellen und des gewünschten Zustands zu berechnen. Die wichtigsten Konstanten sind *Pgain*, *Igain*, *Dgain*, *bias*, *FF0*, *FF1*, *FF2* und *FF3*. Alle diese Parameter müssen einen vernünftigen Wert haben, damit sich der Regler richtig verhält.

Bei der derzeitigen Implementierung der automatischen Abstimmung werden zwei verschiedene Algorithmen verwendet, die über den Tune-Type-Pin ausgewählt werden. Ist der Tune-Type gleich Null, wirkt er sich auf *Pgain*, *Igain* und *Dgain* aus, während *FF0*, *FF1* und *FF2* auf Null gesetzt werden. Ist der Tune-Type 1, wirkt er sich auf *Pgain*, *Igain* und *FF1* aus, während *Dgain*, *FF0* und *FF2* auf Null gesetzt werden. Hinweis: Bei Typ 1 muss die Skalierung so eingestellt werden, dass die Ausgabe in Benutzereinheiten pro Sekunde erfolgt.

Beim automatischen Tuning eines Motors mit Tune-Typ 0 erzeugt der Algorithmus ein Rechteckwellenmuster, das um den "Bias"-Wert auf dem Ausgangspin des PID-Reglers zentriert ist und sich vom positiven Extrem zum negativen Extrem des Ausgangsbereichs bewegt. Dies kann mit dem HAL Scope von LinuxCNC gesehen werden. Für eine Motorsteuerung, die +-10 V als Steuersignal nimmt, kann dies den Motor für kurze Zeit mit voller Geschwindigkeit in eine Richtung beschleunigen, bevor er angewiesen wird, in die entgegengesetzte Richtung mit voller Geschwindigkeit zu fahren. Stellen Sie sicher, dass auf beiden Seiten der Startposition viel Platz ist, und beginnen Sie mit einem niedrigen "Tune-Effort" -Wert, um die verwendete Geschwindigkeit zu begrenzen. Der Wert *tune-effort* definiert den extremen *output*-Wert, der verwendet wird, wenn also *tune-effort* 1 ist, verschiebt sich der *output* -Wert während des Tunings von 1 auf -1. Mit anderen Worten, die Extreme des Wellenmusters werden durch den "Tune-Effort" -Pin gesteuert. Ein zu hoher "Tune-Effort" kann den Motortreiber überlasten.

Die Anzahl der Zyklen im Tune-Muster wird durch den Pin *tune-cycles* gesteuert. Wenn man versucht, die Richtung eines physikalischen Objekts sofort zu ändern (z. B. indem man direkt von einer positiven Spannung zur entsprechenden negativen Spannung im Fall des Motorcontrollers übergeht), ändert sich die Geschwindigkeit natürlich nicht sofort, und es dauert einige Zeit, bis das Objekt langsamer wird und sich in die entgegengesetzte Richtung bewegt. Dies führt zu einer gleichmäßigeren Wellenform auf dem

Positionsstift, da die betreffende Achse hin und her schwingt. Wenn die Achse die Zielgeschwindigkeit in der entgegengesetzten Richtung erreicht hat, wechselt der Autotuner erneut die Richtung. Nach mehreren dieser Änderungen wird die durchschnittliche Zeitverzögerung zwischen den "Spitzen" und "Tälern" dieser Bewegungskurve verwendet, um die vorgeschlagenen Werte für "Pgain", "Igain" und "Dgain" zu berechnen und sie in das HAL-Modell einzufügen, das vom PID-Regler verwendet wird. Die automatisch abgestimmten Einstellungen sind nicht perfekt, können aber einen guten Ausgangspunkt für die weitere Abstimmung der Parameter bilden.

FIXME: Der Autor dieser Anleitung hat die automatische Abstimmung mit Tune-Type auf 1 nicht getestet, so dass dieser Ansatz noch dokumentiert werden muss.

Mit diesem Wissen bewaffnet, ist es nun an der Zeit, sich anzuschauen, wie man die Einstellung vornimmt. Nehmen wir an, die fragliche HAL-Konfiguration lädt die PID-Komponente für X, Y und Z wie folgt, wobei benannte Pin-Namen anstelle von "count=3" verwendet werden:

```
loadrt pid names=pid.x,pid.y,pid.z
```

Wenn die Komponente stattdessen *count=3* verwendet hätte, müssten alle Verwendungen von pid.x, pid.y und pid.z in pid.1, pid.2 bzw. pid.3 geändert werden. Um mit der Einstellung der X-Achse zu beginnen, bewegen Sie die Achse in die Mitte ihres Bereichs, um sicherzustellen, dass sie nirgendwo anstößt, wenn sie sich hin und her bewegt. Sie wollen auch die Achse ferror Grenze (Schleppfehler) zu verlängern, um LinuxCNC akzeptieren die höhere Positionsabweichung während der Abstimmung zu machen. Die sinnvolle ferror Grenze hängt von der Maschine und Setup, aber 1 Zoll oder 20 mm könnten nützliche Ausgangspunkte sein. Als nächstes setzen Sie den anfänglichen *tune-effort* auf eine niedrige Zahl im Leistungsbereich, z.B. 1/100 der maximalen Leistung, und erhöhen ihn langsam, um genauere Tuningwerte zu erhalten. Weisen Sie dem Wert "tune-mode" den Wert 1 zu. Beachten Sie, dass dadurch der PID-Steuerungsteil deaktiviert wird und der "Bias"-Wert an den Ausgangspin geleitet wird, was zu einer starken Abweichung führen kann. Es könnte eine gute Idee sein, den Motortreiber abzustimmen, um sicherzustellen, dass die Null-Eingangsspannung keine Motordrehung verursacht, oder den *bias*-Wert für den gleichen Effekt anzupassen. Nachdem Sie *tune-mode* eingestellt haben, setzen Sie *tune-start* auf 1, um die automatische Abstimmung zu aktivieren. Wenn alles gut geht, wird Ihre Achse einige Sekunden lang vibrieren und sich hin und her bewegen, und danach werden die neuen Werte für Pgain, Igain und Dgain aktiv sein. Um sie zu testen, ändern Sie *tune-mode* wieder auf 0. Beachten Sie, dass das Zurücksetzen von *tune-mode* auf Null zu einem plötzlichen Ruckeln der Achse führen kann, wenn die Achse wieder in die Sollposition gebracht wird, von der sie während des Tunings möglicherweise abgedriftet ist. Zusammenfassend lässt sich sagen, dass dies die halcmd-Anweisungen sind, die Sie für die automatische Abstimmung eingeben müssen:

```
setp pid.x.tune-effort 0.1
setp pid.x.tune-mode 1
setp pid.x.tune-start 1
# warten, bis die Abstimmung abgeschlossen ist
setp pid.x.tune-mode 0
```

Ein Skript zu helfen, die automatische Abstimmung ist in der LinuxCNC-Code-Repository als *scripts/run-auto-pid-tuner* zur Verfügung gestellt. Dies wird sicherstellen, dass die Maschine eingeschaltet ist und bereit zu laufen, referenziert alle Achsen, wenn es nicht bereits getan ist, überprüfen Sie, ob die zusätzlichen Tuning-Pins verfügbar sind, bewegen Sie die Achse zu seinem Mittelpunkt, führen Sie die

Auto-Tuning durch und wieder aktivieren Sie die Pid-Controller, wenn es fertig ist. Dieser Vorgang kann mehrmals wiederholt werden.

9.7. Neuzuordnung (engl. remap) für das Erweitern von G-Code

9.7.1. Einführung: Erweiterung des RS274NGC-Interpreters durch Remapping von Codes

Eine Definition: Neuzuordnung von Codes

Mit "Neuzuordnung" (engl. Remapping) von Codes meinen wir eine der folgenden Optionen:

1. Definition der Semantik neuer - d.h. derzeit nicht zugewiesener - M- oder G-Codes
2. Definieren Sie die Semantik eines - derzeit begrenzten - Satzes bestehender Codes neu.

Warum sollten Sie den RS274NGC Interpreter erweitern?

Der Satz von Codes (M,G,T,S,F), die derzeit vom RS274NGC-Interpreter verstanden werden, ist festgelegt und kann nicht durch Konfigurationsoptionen erweitert werden.

Insbesondere implementieren einige dieser Codes eine feste Abfolge von Schritten, die ausgeführt werden müssen. Während einige dieser Codes, wie M6, durch die Aktivierung oder das Überspringen einiger dieser Schritte über INI-Dateioptionen einigermaßen konfiguriert werden können, ist das Verhalten insgesamt ziemlich starr. Wenn Sie also mit dieser Situation zufrieden sind, dann ist dieser Abschnitt des Handbuchs nichts für Sie.

In vielen Fällen bedeutet dies, dass die Unterstützung von Konfigurationen oder Maschinen, die nicht "out of the box" sind, entweder umständlich oder unmöglich ist, oder dass Änderungen auf der Ebene der Sprache "C/C+\+" vorgenommen werden müssen. Letzteres ist aus guten Gründen unpopulär - die Änderung von Interna erfordert ein tiefes Verständnis der Interpreter-Interna und bringt darüber hinaus eine Reihe von Support-Problemen mit sich. Obwohl es denkbar ist, dass bestimmte Patches ihren Weg in die Hauptdistribution von LinuxCNC finden, ist das Ergebnis dieses Ansatzes ein Sammelsurium von Speziallösungen.

Ein gutes Beispiel für diesen Mangel ist die Werkzeugwechselunterstützung in LinuxCNC: Während zufällige Werkzeugwechsler gut unterstützt werden, ist es nahezu unmöglich, eine Konfiguration für eine manuelle Werkzeugwechselmaschine vernünftig zu definieren, wobei beispielsweise ein automatischer Werkzeuglängen-Offset-Schalter nach einem Werkzeugwechsel besucht und entsprechende Offsets gesetzt werden. Auch wenn ein Patch für einen sehr spezifischen Rack-Werkzeugwechsler existiert, hat er nicht seinen Weg zurück in das primäre Quellcode Repository gefunden.

Viele dieser Probleme können jedoch durch die Verwendung einer O-Wort-Prozedur anstelle eines eingebauten Codes behoben werden. Wann immer der - unzureichende - eingebaute Code ausgeführt werden soll, rufen Sie stattdessen die O-Wort-Prozedur auf. Diese Vorgehen ist zwar möglich, aber umständlich - es erfordert eine Quelltextbearbeitung der NGC-Programme, wobei alle Aufrufe des mangelhaften Codes durch einen Aufruf einer O-Wort-Prozedur ersetzt werden müssen.

In seiner einfachsten Form ist ein remapped Code nicht viel mehr als ein spontaner Aufruf einer O-Wort-Prozedur. Dies geschieht hinter den Kulissen - die Prozedur ist auf der Konfigurationsebene sichtbar, aber nicht auf der NGC-Programmebene.

Im Allgemeinen kann das Verhalten eines umgewandelten Codes wie folgt definiert werden:

- Sie definieren eine O-Wort-Unterroutine, die das gewünschte Verhalten implementiert
- Alternativ können Sie auch eine Python-Funktion verwenden, die das Verhalten des Interpreters erweitert.

Wie man Dinge zusammenbringt

M- und G-Codes und O-Wörter Unterprogrammaufrufe haben eine recht unterschiedliche Syntax.

O-Wort-Prozeduren zum Beispiel nehmen Positionsparameter mit einer bestimmten Syntax wie folgt:

```
o<test> call [1.234] [4.65]
```

während M- oder G-Codes in der Regel erforderliche oder optionale "Wort"-Parameter enthalten. Für G76 (Einfädeln) sind beispielsweise die Wörter P, Z, I, J und K erforderlich, und optional sind die Wörter R, Q, H, E und L erforderlich.

Es reicht also nicht aus, einfach zu sagen: "Wann immer Sie auf Code X stoßen, rufen Sie bitte Prozedur Y auf" - es muss zumindest eine Überprüfung und Konvertierung der Parameter stattfinden. Dies erfordert einen "Glue Code" zwischen dem neuen Code und der entsprechenden NGC-Prozedur, der ausgeführt werden muss, bevor die Kontrolle an die NGC-Prozedur übergeben wird.

Dieser "Glue"-Code kann nicht als O-Wort-Prozedur geschrieben werden, da der RS274NGC-Sprache die introspektiven Fähigkeiten und der Zugriff auf interne Datenstrukturen des Interpreters fehlen, um den gewünschten Effekt zu erzielen. Den Glue-Code in - wiederum - *C/C*+|+ zu schreiben, wäre eine unflexible und daher unbefriedigende Lösung.

Wie sich Embedded Python einfügt

Um eine einfache Situation einfach und eine komplexe Situation lösbar zu machen, wird das Problem des Glue Codes als Zwischenebene wie folgt angegangen:

- Für einfache Situationen gibt es eine eingebaute Glue-Prozedur (`argspec`), welche die häufigsten Anforderungen an die Parameterübergabe abdeckt.
- Für die Neuzuordnung von T,M6,M61,S,F gibt es einen Standard-Python-Glue (engl. für Kleber), der die meisten Situationen abdecken sollte, siehe Standard Glue.
- Für komplexere Situationen kann man einen eigenen Python-Glue schreiben, um neues Verhalten zu implementieren.

Die eingebetteten Python-Funktionen im Interpreter waren ursprünglich als Glue-Code gedacht, erwiesen sich aber weit darüber hinaus als sehr nützlich. Benutzer, die mit Python vertraut sind, werden es wahrscheinlich einfacher finden, remapped Codes, Glue, O-Wort-Prozeduren usw. in reinem Python zu schreiben, ohne auf die etwas schwerfällige RS274NGC-Sprache zurückgreifen zu müssen.

Ein Wort zu eingebettetem Python

Viele Menschen sind mit der *Erweiterung* des Python-Interpreters durch *C/C+|+*-Module vertraut, und dies wird in LinuxCNC stark genutzt, um von Python-Skripten aus auf Task-, HAL- und Interpreter-Interna zuzugreifen. *Python erweitern* bedeutet im Grunde: Ihr Python-Skript wird so ausgeführt, als *wäre es der Bestimmer* und kann auf Nicht-Python-Code zugreifen, indem es Erweiterungsmodule importiert und verwendet, die in *C/C+|+* geschrieben sind. Beispiele hierfür sind die LinuxCNC-Module `hal`, `gcode` und `emc`.

Eingebettetes Python ist ein wenig anders und weniger bekannt: Das Hauptprogramm ist in C/C `geschrieben und kann Python wie ein Unterprogramm verwenden. Dies ist ein leistungsfähiger Erweiterungsmechanismus und die Grundlage für die "Skripterweiterungen", die in vielen erfolgreichen Softwarepaketen zu finden sind. Eingebetteter Python-Code kann auf "C/C`"-Variablen und -Funktionen über eine ähnliche Erweiterungsmodulmethode zugreifen.

9.7.2. Erste Schritte

Die Definition eines Codes umfasst die folgenden Schritte:

- Wählen Sie einen Code - verwenden Sie entweder einen nicht zugewiesenen Code oder definieren Sie einen vorhandenen Code neu.
- Entscheiden Sie, wie Parameter gehandhabt werden.
- Entscheiden Sie, ob und wie die Ergebnisse behandelt werden.
- Entscheiden Sie über die Reihenfolge der Ausführung.

Integrierte Neuzuordnungen

Bitte beachten Sie, dass derzeit nur einige bestehende Codes umdefiniert werden können, während es viele "freie" Codes gibt, die für eine Neuzuordnung zur Verfügung stehen. Bei der Entwicklung eines umdefinierten bestehenden Codes ist es eine gute Idee, mit einem nicht zugewiesenen G- oder M-Code zu beginnen, damit Sie sowohl ein bestehendes als auch ein neues Verhalten verwenden können. Wenn Sie fertig sind, definieren Sie den vorhandenen Code so um, dass er Ihre Konfiguration für die Neuzuordnung verwendet.

- Der aktuelle Satz unbenutzter M-Codes, die für die Definition durch den Benutzer zur Verfügung stehen, ist in dem Abschnitt zu unbelegte M-codes zu finden.
- Informationen zu unbelegten G-Codes finden Sie hier.
- Vorhandene Codes, die neu zugewiesen werden können, sind im Abschnitt zu neu zuweisbaren Codes aufgeführt.

Derzeit gibt es zwei vollständige, nur in Python verfügbare Remaps, die in stdglue.py verfügbar sind:

- ignore_m6
- index_lathe_tool_with_wear

Diese sind für die Verwendung mit Drehmaschinen gedacht. Drehbänke verwenden nicht M6, um die Werkzeuge zu indexieren, sondern den Befehl T.

Diese Neuzuordnung fügt auch Verschleißkorrekturen zur Werkzeugkorrektur hinzu, d.h. T201 würde auf Werkzeug 2 indexiert (mit der Werkzeugkorrektur von Werkzeug 2) und fügt die Verschleißkorrektur 1 hinzu. In der Werkzeugtabelle sind die Werkzeugnummern über 10000 Verschleißkorrekturen, d.h. in der Werkzeugtabelle wäre das Werkzeug 10001 die Verschleißkorrektur 1.

Hier ist, was Sie in der INI brauchen, um sie zu verwenden:

```
[RS274NGC]
REMAP=T python=index_lathe_tool_with_wear
REMAP=M6 python=ignore_m6

[PYTHON]
# where to find the Python code:

# Code spezifisch für diese Konfiguration
PATH_PREPEND=./

# generischer Support-Code - stellen Sie sicher, dass dieser tatsächlich auf Python-
stdglue zeigt
PATH_APPEND=../../nc_files/remap_lib/python-stdglue/

# importieren Sie das folgende Python-Modul
TOPLEVEL=toplevel.py

# je höher, desto ausführlicher die Aufzeichnung des Python-Plugins
LOG_LEVEL = 0
```

Sie müssen auch die erforderliche Python-Datei in Ihrem Konfigurationsordner hinzufügen.

Upgrade einer bestehenden Konfiguration

Auswahl eines Codes

Beachten Sie, dass derzeit nur einige wenige bestehende Codes umdefiniert werden können, während es viele "freie" Codes gibt, die durch eine Neuzuordnung verfügbar gemacht werden könnten. Bei der Entwicklung eines umdefinierten bestehenden Codes ist es sinnvoll, mit einem nicht zugewiesenen G- oder M-Code zu beginnen, damit sowohl das bestehende als auch das neue Verhalten geübt werden kann. Wenn Sie fertig sind, definieren Sie den bestehenden Code neu, um Ihre Remapping-Einstellung zu verwenden.

- Die aktuelle Menge der nicht verwendeten M-Codes, die vom Benutzer definiert werden können, finden Sie hier.
- Nicht zugeordnete G-Codes werden hier aufgelistet.
- Vorhandene Codes, die neu zugeordnet werden können, sind in dieser Liste aufgeführt.

Handhabung der Parameter

Nehmen wir an, der neue Code wird durch eine NGC-Prozedur definiert und benötigt einige Parameter, von denen einige erforderlich und andere optional sein können. Wir haben die folgenden Optionen, um der Prozedur Werte zuzuführen:

1. Extraktion von Wörtern aus dem aktuellen Block und Übergabe an die Prozedur als Parameter (z.B. `X22.34` oder `P47`),
2. unter Bezugnahme auf INI-Datei-Variablen,
3. Bezugnahme auf globale Variablen (wie `#2200 = 47.11` oder `#<_global_param> = 315.2`).

Die erste Methode wird für dynamische Parameter wie Positionen bevorzugt. Sie müssen definieren, welche Wörter des aktuellen Blocks eine Bedeutung für Ihren neuen Code haben, und angeben, wie diese an die NGC-Prozedur übergeben werden. Ein einfacher Weg ist die Verwendung der argspec-Anweisung. Ein eigener Prolog könnte bessere Fehlermeldungen liefern.

Die Verwendung von INI-Datei-Variablen ist besonders nützlich, wenn Sie sich auf Einrichtungsinformationen für Ihre Maschine beziehen, z. B. auf eine feste Position wie die Position eines Werkzeuglängensensors. Der Vorteil dieser Methode ist, dass die Parameter für Ihre Konfiguration festgelegt sind, unabhängig davon, welche NGC-Datei Sie gerade ausführen.

Es ist immer möglich, auf globale Variablen zu verweisen, aber sie werden leicht übersehen.

Beachten Sie, dass es nur eine begrenzte Anzahl von Wörtern gibt, die als Parameter verwendet werden können, so dass man möglicherweise auf die zweite und dritte Methode zurückgreifen muss, wenn viele Parameter benötigt werden.

Handhabung der Ergebnisse

Ihr neuer Code kann erfolgreich sein oder scheitern, z. B. wenn eine ungültige Parameterkombination übergeben wird. Oder Sie entscheiden sich dafür, die Prozedur "einfach auszuführen" und die Ergebnisse zu ignorieren, in diesem Fall gibt es nicht viel Arbeit zu tun.

Epilog-Handler helfen bei der Verarbeitung der Ergebnisse von Remap-Prozeduren - siehe den Referenzabschnitt.

Ausführungsreihenfolge

Ausführbare G-Code-Wörter werden in Modalgruppen eingeteilt, was auch ihr relatives Ausführungsverhalten definiert.

Wenn ein G-Code-Block mehrere ausführbare Wörter in einer Zeile enthält, werden diese Wörter in einer vordefinierten Ausführungsreihenfolge ausgeführt, nicht in der Reihenfolge, in der sie im Block erscheinen.

Wenn Sie einen neuen ausführbaren Code definieren, weiß der Interpreter noch nicht, wo Ihr Code in dieses Schema passt. Aus diesem Grund müssen Sie eine geeignete Modalgruppe wählen, in der Ihr Code ausgeführt werden soll.

Ein minimales Beispiel für neu zugeordneten Code

Damit Sie sich ein Bild davon machen können, wie die einzelnen Teile zusammenpassen, wollen wir eine ziemlich minimale, aber vollständige Definition von neu zugeordnetem Code untersuchen. Wir wählen einen nicht zugewiesenen M-Code und fügen die folgende Option zur INI-Datei hinzu:

```
[RS274NGC]
REMAP=M400  modalgroup=10 argspec=Pq ngc=myprocedure
```

Zusammengefasst bedeutet dies:

- Der M400-Code hat einen erforderlichen Parameter P und einen optionalen Parameter Q. Andere Wörter im aktuellen Block werden in Bezug auf den M400-Code ignoriert. Wenn das Wort P nicht vorhanden ist, schlägt die Ausführung mit einem Fehler fehl.
- Wenn ein M400-Code auftritt, wird myprocedure.ngc zusammen mit den anderen modal group 10 M-Codes gemäß der Ausführungsreihenfolge ausgeführt.
- Der Wert von "P" und "Q" sind in der Prozedur als lokale benannte Parameter verfügbar. Sie können als #<P> und #<Q> bezeichnet werden. Die Prozedur kann testen, ob das Wort Q mit der eingebauten Funktion EXISTS vorhanden war.

Es wird erwartet, dass die Datei myprocedure.ngc im Verzeichnis [DISPLAY]NC_FILES oder [RS274NGC]SUBROUTINE_PATH existiert.

Eine ausführliche Erläuterung der REMAP (engl. für Neuzuordnung)-Parameter finden Sie im folgenden Referenzteil.

9.7.3. Neuzuordnung konfigurieren

Die REMAP-Anweisung

Um einen Code neu zuzuordnen, definieren Sie ihn mit der Option REMAP im Abschnitt RS274NG Ihrer INI-Datei. Verwenden Sie eine REMAP-Zeile pro neu zugeordnetem Code.

Die Syntax von *REMAP* lautet:

`REMAP=`<*code*> <*options*>

wobei <*code*> einer der Codes T, M6, M61, S, F (bestehende Codes) oder einer der nicht zugewiesenen M-Codes oder G-Codes sein kann.

Es ist ein Fehler, den Parameter <*code*> wegzulassen.

Die Optionen der REMAP-Anweisung werden durch Leerzeichen getrennt. Die Optionen sind Schlüsselwort-Wert-Paare und lauten derzeit:

`modalgroup=`<*modal group*>

G-Codes

Die einzige derzeit unterstützte modale Gruppe ist 1, die auch der Standardwert ist, wenn keine Gruppe angegeben wird. Gruppe 1 bedeutet "neben anderen G-Codes ausführen".

M-Codes

Die derzeit unterstützten Modalgruppen sind: 5,6,7,8,9,10. Wird keine Modalgruppe angegeben, wird standardmäßig 10 ("nach allen anderen Wörtern des Blocks ausführen") verwendet.

T,S,F

für diese ist die modale Gruppe festgelegt und die Option `modalgroup=` wird ignoriert.

`argspec=`<*argspec*>

Siehe Beschreibung der argspec-Parameteroptionen. Optional.

`ngc=`<*ngc_basename*>

Basisname des Dateinamens eines O-Wort-Unterprogramms. Die Erweiterung .ngc darf nicht angegeben werden. Gesucht wird in den Verzeichnissen, die in dem in `[DISPLAY]PROGRAM_PREFIX` angegebenen Verzeichnis angegeben sind, dann in `[RS274NGC]SUBROUTINE_PATH`. Wechselseitig exklusiv mit `python=`. Es ist ein Fehler, sowohl `ngc=` als auch `python=` wegzulassen.

`python=`<*Python function name*>

Anstatt eine ngc O-Wort-Prozedur aufzurufen, rufen Sie eine Python-Funktion auf. Es wird erwartet, dass die Funktion im Modul `module_basename.oword` definiert ist. Wechselseitig exklusiv mit `ngc=`.

`prolog=`<*Python function name*>

Bevor Sie eine ngc-Prozedur ausführen, rufen Sie diese Python-Funktion auf. Es wird erwartet, dass die Funktion in dem Modul `module_basename.remap` definiert ist. Optional.

`epilog=`<*Python function name*>

Nach der Ausführung einer ngc-Prozedur rufen Sie diese Python-Funktion auf. Es wird erwartet, dass die Funktion in dem Modul `module_basename.remap` definiert ist. Optional.

Die Optionen `python`, `prolog` und `epilog` erfordern, dass das Python-Interpreter-Plugin configured ist und dass dort entsprechende Python-Funktionen definiert sind, damit sie mit diesen Optionen angesprochen werden können.

Die Syntax für die Definition eines neuen Codes und die Umdefinierung eines bestehenden Codes sind identisch.

Nützliche REMAP-Optionskombinationen

Beachten Sie, dass zwar viele Kombinationen von argspec-Optionen möglich sind, aber nicht alle von ihnen sinnvoll sind. Die folgenden Kombinationen sind nützliche Idiome:

`argspec=`<*words*> `ngc=`<*procname*> `modalgroup=`_<*group*>

Der empfohlene Weg, eine NGC-Prozedur mit einer Standard-Argspec-Parameterumwandlung aufzurufen. Wird verwendet, wenn argspec gut genug ist. Beachten Sie, dass dies für die Neuzuordnung der `T`__x__- und `M6`/`M61`-Werkzeugwechselcodes nicht ausreicht.

`prolog=`<*pythonprolog*> `ngc=`<*procname*> `epilog=`<*pythonepilog*> `modalgroup=`<*group*>

Rufen Sie eine Python-Prolog-Funktion auf, um alle vorbereitenden Schritte durchzuführen, und rufen Sie dann die NGC-Prozedur auf. Danach rufen Sie die Python-Epilog-Funktion auf, um alle Aufräumarbeiten oder die Extraktion von Ergebnissen durchzuführen, die nicht im G-Code behandelt werden können. Dies ist der flexibelste Weg, einen Code in eine NGC-Prozedur umzuwandeln, da auf fast alle internen Variablen des Interpreters und einige interne Funktionen von den Prolog- und Epilog-Handlern aus zugegriffen werden kann. Außerdem hat man ein längeres Seil, an dem man

sich aufhängen kann.

`python=`<*pythonfunction*> `modalgroup=`<*group*>

Direkter Aufruf einer Python-Funktion ohne Argumentumwandlung. Die leistungsfähigste Art, einen Code umzuwandeln und direkt zu Python zu wechseln. Verwenden Sie dies, wenn Sie keine NGC-Prozedur benötigen oder NGC Ihnen nur im Weg steht.

`argspec=`<*words*> `python=`<*pythonfunction*> `modalgroup=`<*group*>

Konvertiert die argspec-Wörter und übergibt sie an eine Python-Funktion als Schlüsselwort-Argument-Wörterbuch. Verwenden Sie dies, wenn Sie zu faul sind, die im Block übergebenen Wörter selbst zu untersuchen.

Wenn Sie lediglich Python-Code aus G-Code aufrufen wollen, gibt es den etwas einfacheren Weg Aufruf von Python-Funktionen wie O-word-Prozeduren.

Der argspec-Parameter

Die Argumentenspezifikation (Schlüsselwort `argspec`) beschreibt erforderliche und optionale Wörter, die an eine ngc-Prozedur übergeben werden, sowie optionale Vorbedingungen für die Ausführung dieses Codes.

Ein argspec besteht aus 0 oder mehr Zeichen der Klasse `[@A-KMNP-Za-kmnp-z^>]`. Er kann leer sein (wie `argspec=`).

Ein leeres argspec oder gar kein argspec-Argument bedeutet, dass der umgewandelte Code keine Parameter von dem Block erhält. Eventuell vorhandene zusätzliche Parameter werden ignoriert.

Beachten Sie, dass die RS274NGC-Regeln weiterhin gelten - zum Beispiel dürfen Sie Achsenwörter (z. B. `X`, `Y`, `Z`) nur im Zusammenhang mit einem G-Code verwenden.

Achsenwörter können auch nur verwendet werden, wenn die Achse aktiviert ist. Wenn nur XYZ aktiviert ist, kann ABCUVW nicht in argspec verwendet werden.

Die Wörter `FST? haben die normalen Funktionen, sind aber als Variablen in der neu zugeordneten Funktion verfügbar. `F` setzt den Vorschub, `S` setzt die Spindeldrehzahl, `T` löst die Werkzeugvorbereitungsfunktion aus. Die Wörter `FST` sollten nicht verwendet werden, wenn dieses Verhalten nicht erwünscht ist.

Die Wörter `DEIJKPQR` haben keine vordefinierte Funktion und werden für die Verwendung als argspec-Parameter empfohlen.

`ABCDEFHIJKPQRSTUVWXYZ`

Definiert einen erforderlichen Wortparameter: ein Großbuchstabe gibt an, dass das entsprechende Wort im aktuellen Block vorhanden sein **muss**. Der Wert des Wortes wird als lokaler benannter Parameter mit einem entsprechenden Namen übergeben. Wenn das Zeichen "@" in der Argspec vorhanden ist, wird es als Positionsparameter übergeben, siehe unten.

`abcdefhijkpqrstuvwxyz`

Definiert einen optionalen Wortparameter: ein Kleinbuchstabe gibt an, dass das entsprechende Wort

im aktuellen Block vorhanden sein **kann**. Wenn das Wort vorhanden ist, wird der Wert des Wortes als lokaler benannter Parameter übergeben. Wenn das Zeichen "@" in der Argspec vorhanden ist, wird es als Positionsparameter übergeben, siehe unten.

`@`

Das `@` (at-Zeichen, Klammeraffe) weist argspec an, Wörter als Positionsparameter zu übergeben, und zwar in der Reihenfolge, die nach der `@`-Option definiert ist. Beachten Sie, dass bei der Übergabe von Positionsparametern eine Prozedur nicht erkennen kann, ob ein Wort vorhanden war oder nicht, siehe Beispiel unten.

TIP dies hilft bei der Paketierung bestehender NGC-Prozeduren als remapped codes. Vorhandene Prozeduren erwarten positionale Parameter. Mit der Option "@" können Sie vermeiden, dass sie umgeschrieben werden, um auf lokale benannte Parameter zu verweisen.

`^`

Das Zeichen `^` (Dach, Zirkumflex, engl. caret) gibt an, dass die aktuelle Spindeldrehzahl größer als Null sein muss (Spindel läuft), sonst schlägt der Code mit einer entsprechenden Fehlermeldung fehl.

`>`

Das Zeichen `>` (größer-als) gibt an, dass der aktuelle Vorschub größer als Null sein muss, andernfalls schlägt der Code mit einer entsprechenden Fehlermeldung fehl.

`n`

Das `n` (größer als) Zeichen gibt an, dass die aktuelle Zeilennummer im `n`lokal benannten Parameter übergeben wird.

Standardmäßig werden Parameter als lokale benannte Parameter an eine NGC-Prozedur übergeben. Diese lokalen Parameter erscheinen als "bereits gesetzt", wenn die Prozedur mit der Ausführung beginnt, was sich von der bestehenden Semantik unterscheidet (lokale Variablen beginnen mit dem Wert 0.0 und müssen explizit mit einem Wert versehen werden).

Optionale Wortparameter können mit dem Idiom `EXISTS(#<Wort>)` auf ihr Vorhandensein getestet werden.

Beispiel für die Übergabe von benannten Parametern an NGC-Prozeduren

Angenommen, der Code ist wie folgt definiert

```
REMAP=M400 modalgroup=10 argspec=Pq ngc=m400
```

und `m400.ngc` sieht wie folgt aus:

```
o<m400> sub
(P is required since it is uppercase in the argspec)
(debug, P word=#<P>)
(the q argspec is optional since its lowercase in the argspec. Use as follows:)
o100 if [EXISTS[#<q>]]
    (debug, Q word set: #<q>)
o100 endif
```

```
o<m400> endsub
M2
```

- Die Ausführung von `M400` wird fehlschlagen mit der Meldung `user-defined M400: missing: P`.
- Die Ausführung von `M400 P123` wird `P-Wort=123.000000` anzeigen.
- Die Ausführung von `M400 P123 Q456` zeigt `P-Wort=123.000000` und `Q-Wortsatz: 456.000000`.

Beispiel für die Übergabe von Positionsparametern an NGC-Prozeduren

Angenommen, der Code ist wie folgt definiert

```
REMAP=M410 modalgroup=10 argspec=@PQr ngc=m410
```

und `m400.ngc` sieht wie folgt aus:

```
o<m410> sub
(debug, [1]=#1 [2]=#2 [3]=#3)
o<m410> endsub
M2
```

- Bei Ausführung von `M410 P10` wird angezeigt: `m410.ngc: [1]=10.000000 [2]=0.000000`.
- Bei der Ausführung von `M410 P10` wird angezeigt: `m410.ngc: [1]=10.000000 [2]=0.000000`.

NOTE Sie verlieren die Fähigkeit, mehr als ein optionales Parameterwort zu unterscheiden, und Sie können nicht feststellen, ob ein optionaler Parameter vorhanden war, aber den Wert 0 oder gar nicht vorhanden war.

Einfaches Beispiel für die Übergabe von benannten Parametern an eine Python-Funktion

Es ist möglich, neue Codes *ohne* ein NGC-Verfahren zu definieren. Hier ist ein einfaches erstes Beispiel, ein komplexeres Beispiel finden Sie im nächsten Abschnitt.

Angenommen, der Code ist wie folgt definiert

```
REMAP=G88.6 modalgroup=1 argspec=XYZp python=g886
```

Dies weist den Interpreter an, die Python-Funktion `g886` im Modul `module_basename.remap` auszuführen, was etwa so aussehen könnte:

```
from interpreter import INTERP_OK
from emccanon import MESSAGE

def g886(self, **words):
    for key in words:
        MESSAGE("word '%s' = %f" % (key, words[key]))
    if words.has_key('p'):
        MESSAGE("the P word was present")
    MESSAGE("comment on this line: '%s'" % (self.blocks[self.remap_level].comment))
    return INTERP_OK
```

Probieren Sie dies mit aus mit: g88.6 x1 y2 z3 g88.6 x1 y2 z3 p33 (ein Kommentar hier)

Sie werden die schrittweise Einführung der eingebetteten Python-Umgebung bemerken - siehe hier für Details. Beachten Sie, dass es bei Python-Remapping-Funktionen keinen Sinn macht, Python-Prolog- oder Epilog-Funktionen zu haben, da es sich in erster Linie um die Ausführung einer Python-Funktion handelt.

Erweitertes Beispiel: Neu zugeordnete Codes in reinem Python

Die Module `interpreter` und `emccanon` legen den größten Teil des Interpreters und einige Canon-Interna offen, so dass viele Dinge, die bisher in *C/C+|+* programmiert werden mussten, nun in Python erledigt werden können.

Das folgende Beispiel basiert auf dem Skript `nc_files/involute.py` - aber als G-Code mit einigen Parameterextraktionen und -überprüfungen festgehalten. Es demonstriert auch den rekursiven Aufruf des Interpreters (siehe `self.execute()`).

Angenommen, die Definition lautet wie folgt (Anmerkung: Hier wird argspec nicht verwendet):

```
REMAP=G88.1 modalgroup=1 py=involute
```

Die unten aufgefuehrte Funktion `involute` in `python/remap.py` macht alle Wortextraktionen direkt aus dem aktuellen Block. Beachten Sie, dass Interpreterfehler in Python-Ausnahmen übersetzt werden können. Denken Sie daran, dass es sich hierbei um eine "Vorlaufzeit" handelt - Ausführungszeitfehler können auf diese Weise nicht abgefangen werden.

```
import sys
import traceback
from math import sin,cos

from interpreter import *
from emccanon import MESSAGE
from util import lineno, call_pydevd
# raises InterpreterException if execute() or read() fails
throw_exceptions = 1

def involute(self, **words):
    """ remap-Funktion mit Rohzugriff auf Interpreter-Interna """

    if self.debugmask & 0x20000000: call_pydevd() # USER2 debug flag

    if equal(self.feed_rate,0.0):
        return "feedrate > 0 required"

    if equal(self.speed[0], 0.0):
        return "spindle speed > 0 required"

    plunge = 0.1 # if Z word was given, plunge - with reduced feed

    # Kontrollblock auf relevante Wörter untersuchen
    c = self.blocks[self.remap_level]
    x0 = c.x_number if c.x_flag else 0
    y0 = c.y_number if c.y_flag else 0
    a  = c.p_number if c.p_flag else 10
    old_z = self.current_z
```

```
    if self.debugmask & 0x10000000:
        print("x0=%f y0=%f a=%f old_z=%f" % (x0,y0,a,old_z))

    try:
        #self.execute("G3456")  # would raise InterpreterException
        self.execute("G21",lineno())
        self.execute("G64 P0.001",lineno())
        self.execute("G0 X%f Y%f" % (x0,y0),lineno())

        if c.z_flag:
            feed = self.feed_rate
            self.execute("F%f G1 Z%f" % (feed * plunge, c.z_number),lineno())
            self.execute("F%f" % (feed),lineno())

        for i in range(100):
            t = i/10.
            x = x0 + a * (cos(t) + t * sin(t))
            y = y0 + a * (sin(t) - t * cos(t))
            self.execute("G1 X%f Y%f" % (x,y),lineno())

        if c.z_flag: # retract to starting height
            self.execute("G0 Z%f" % (old_z),lineno())

    except InterpreterException,e:
        msg = "%d: '%s' - %s" % (e.line_number,e.line_text, e.error_message)
  return msg

    return INTERP_OK
```

Die bisher beschriebenen Beispiele finden Sie in "configs/sim/axis/remap/getting-started" mit vollständigen Arbeitskonfigurationen.

9.7.4. Aktualisieren einer bestehenden Konfiguration für die Neuzuordnung

Die Mindestvoraussetzungen für die Verwendung von "REMAP"-Anweisungen sind wie folgt:

- Das Python-Plugin muss durch Angabe eines `[PYTHON]TOPLEVEL=<path-to-toplevel-script>` in der INI-Datei aktiviert werden.
- Das Toplevel-Skript muss das Modul `remap` importieren, das anfangs leer sein kann, aber der Import muss vorhanden sein.
- Der Python-Interpreter muss das obige remap.py-Modul finden, daher muss der Pfad zu dem Verzeichnis, in dem sich Ihre Python-Module befinden, mit `[PYTHON]PATH_APPEND=<Pfad-zu-Ihrem-Lokalen-Python-Verzeichnis>` hinzugefügt werden
- Empfohlen: Importieren Sie die `stdglue` Handler im `remap` Modul. In diesem Fall muss Python auch `stdglue.py` finden - wir kopieren es einfach aus der Distribution, damit Sie bei Bedarf lokale Änderungen vornehmen können. Abhängig von Ihrer Installation kann der Pfad zu `stdglue.py` variieren.

Angenommen, Ihre Konfiguration befindet sich unter `/home/user/xxx` und die INI-Datei lautet `/home/user/xxx/xxx.ini`, führen Sie die folgenden Befehle aus.

```
$ cd /home/user/xxx
$ mkdir python
$ cd python
$ cp /usr/share/linuxcnc/ncfiles/remap_lib/python-stdglue/stdglue.py .
$ echo 'from stdglue import *' >remap.py
$ echo 'import remap' >toplevel.py
```

Editieren Sie nun ``/home/user/``*xxx*``/``*xxx*``.ini`` und fügen folgendes hinzu:

```
[PYTHON]
TOPLEVEL=/home/user/xxx/python/toplevel.py
PATH_APPEND=/home/user/xxx/python
```

Überprüfen Sie nun, dass LinuxCNC ohne Fehlermeldungen hochkommt - führen Sie es in einem Terminalfenster aus:

```
$ cd /home/user/xxx
$ linuxcnc xxx.ini
```

9.7.5. Codes für den Wechsel des Remapping-Werkzeugs: T, M6, M61

Übersicht

Wenn Sie mit den Interna von LinuxCNC nicht vertraut sind, lesen Sie zuerst den Abschnitt How tool change currently works (dire but necessary).

Beachten Sie, dass wir bei der Neuzuordnung eines bestehenden Codes die this codes' built-in functionality des Interpreters vollständig deaktivieren.

Unser remapped Code muss also etwas mehr tun, als nur einige Befehle zu generieren, um die Maschine so zu bewegen, wie wir es wollen - er muss auch die Schritte aus dieser Sequenz wiederholen, die nötig sind, um den Interpreter und die `Task` bei Laune zu halten.

Dies hat jedoch **keine** Auswirkungen auf die Verarbeitung von Befehlen, die sich auf Werkzeugwechsel in `task` und `iocontrol` beziehen. Das heißt, wenn wir step 6b ausführen, wird dies immer noch iocontrol auslösen.

Entscheidungen, Entscheidungen:

- Möchten wir eine O-Wort-Prozedur verwenden oder alles in Python-Code tun?
- Ist die "iocontrol"-HAL-Sequenz (tool-prepare/tool-prepared und tool-change/tool-changed Pins) gut genug oder brauchen wir eine andere Art von HAL-Interaktion für unseren Werkzeugwechsler (z.B.: mehr beteiligte HAL-Pins mit einer anderen Interaktionssequenz)?

Je nach Antwort ergeben sich vier verschiedene Szenarien:

- Wenn wir eine O-Wort-Prozedur verwenden, benötigen wir Prolog- und Epilog-Funktionen.
- wenn nur Python-Code und keine O-Wort-Prozedur verwendet wird, genügt eine Python-Funktion.

- Bei Verwendung der *iocontrol*-Pins enthält unsere O-Wort-Prozedur oder unser Python-Code hauptsächlich Bewegungen.
- Wenn wir eine komplexere Interaktion als die von `iocontrol` angebotene benötigen, müssen wir unsere eigene Interaktion vollständig definieren, indem wir `motion.digital*` und `motion.analog*` Pins verwenden und die `iocontrol` Pins im Wesentlichen ignorieren, indem wir sie in eine Schleife schalten.

NOTE Wenn Sie O-Wort-Prozeduren hassen und Python lieben, steht es Ihnen frei, alles in Python zu machen. In diesem Fall würden Sie einfach eine `python=`<*Funktion*> -Spezifikation in der REMAP-Anweisung haben. Aber da wir davon ausgehen, dass die meisten Leute an der Verwendung von O-Wort-Prozeduren interessiert sind, weil sie damit vertrauter sind, werden wir das als erstes Beispiel verwenden.

Der Gesamtansatz für unser erstes Beispiel lautet also:

1. Wir würden gerne so viel wie möglich mit G-Code in einer O-Wort-Prozedur machen, um flexibel zu sein. Das schließt alle HAL-Interaktionen ein, die normalerweise von `iocontrol` gehandhabt werden - weil wir lieber clevere Dinge mit Moves, Probes, HAL-Pin I/O und so weiter machen wollen.
2. Wir werden versuchen, den Python-Code auf das notwendige Maß zu reduzieren, um den Interpreter zufrieden zu stellen und `task` dazu zu bringen, tatsächlich etwas zu tun. Das wird in den Python-Funktionen "prolog" und "epilog" geschehen.

Verstehen der Rolle von "iocontrol" mit neu zugeordneten Werkzeugwechselcodes

`iocontrol` bietet zwei HAL-Interaktionssequenzen, die wir verwenden oder nicht verwenden können:

- Wenn die durch einen SELECT_TOOL()-Kanonbefehl in die Warteschlange gestellte NML-Nachricht ausgeführt wird, löst dies neben dem Setzen der *XXXX*-Pins die HAL-Sequenz "Werkzeug vorbereiten und warten, bis Werkzeug vorbereitet hoch wird" in `iocontrol` aus
- Wenn die NML-Nachricht, die durch den Kanon-Befehl CHANGE_TOOL() in die Warteschlange gestellt wurde, ausgeführt wird, löst dies die HAL-Sequenz "raise tool-change and wait for tool-changed to become high" (Werkzeugwechsel auslösen und darauf warten, dass das Werkzeug geändert wird) in `iocontrol` aus und setzt außerdem die *XXXX*-Pins

Sie müssen entscheiden, ob die vorhandenen "iocontrol"-HAL-Sequenzen ausreichen, um Ihren Wechsler zu steuern. Vielleicht brauchen Sie eine andere Interaktionssequenz - zum Beispiel mehr HAL-Pins oder vielleicht eine komplexere Interaktion. Je nach Antwort können wir die vorhandenen `iocontrol`-HAL-Sequenzen weiter verwenden oder unsere eigenen definieren.

Aus Dokumentationsgründen werden wir diese `iocontrol`-Sequenzen deaktivieren und unsere eigenen Sequenzen erstellen - das Ergebnis wird wie die bestehende Interaktion aussehen und sich auch so anfühlen, aber jetzt haben wir die vollständige Kontrolle über sie, da sie in unserer eigenen O-Wort-Prozedur ausgeführt werden.

Wir werden also einige `motion.digital-*` und `motion.analog-*` Pins und die zugehörigen `M62` ... `M68` Befehle verwenden, um unsere eigene HAL Interaktion in unserer O-Wort Prozedur durchzuführen, und diese werden effektiv die `iocontrol` *tool-prepare/tool-prepared* und *tool-change/tool-changed*

Sequenzen ersetzen. Wir definieren also unsere Pins, welche die vorhandenen "iocontrol"-Pins funktionell ersetzen, und machen aus den "iocontrol"-Interaktionen eine Schleife. In unserem Beispiel werden wir die folgende Korrespondenz verwenden:

Entsprechung der "iocontrol"-Pins in den Beispielen

`iocontrol.0` **pin**	`motion` **pin**
`tool-prepare`	`digital-out-00`
`tool-prepared`	`digital-in-00`
`tool-change`	`digital-out-01`
`tool-changed`	`digital-in-01`
`tool-prep-number`	`analog-out-00`
`tool-prep-pocket`	`analog-out-01`
`tool-number`	`analog-out-02`

Nehmen wir an, Sie wollen den M6-Befehl umdefinieren und durch eine O-Wort-Prozedur ersetzen, aber ansonsten sollte alles "weiter funktionieren".

Unser O-Wort-Verfahren würde also die Schritte hier ersetzen. Wenn Sie sich diese Schritte ansehen, werden Sie feststellen, dass NGC-Code für die meisten, aber nicht für alle, verwendet werden kann. Das, was NGC nicht kann, wird in Python Prolog- und Epilog-Funktionen erledigt.

Spezifikation des M6-Ersatzes

Um die Idee zu vermitteln, ersetzen wir einfach die eingebaute M6-Semantik durch unsere eigene. Sobald das funktioniert, können Sie alle Aktionen, die Sie für sinnvoll halten, in die O-Wort-Prozedur einfügen.

Wenn wir die Schritte durchlaufen, finden wir:

1. prüfen, ob der T-Befehl bereits ausgeführt wurde - **Ausführung im Python-Prolog**
2. Prüfen, ob der Schneidwerksausgleich aktiv ist - **Ausführung in Python-Prolog**
3. Anhalten der Spindel, falls erforderlich - **kann in NGC durchgeführt werden**
4. nach oben fahren - **kann in NGC vorgenommen werden**
5. wenn TOOL_CHANGE_AT_G30 gesetzt wurde:
   a. Verschieben Sie die A-, B- und C-Indexer, falls zutreffend - **NGC kann das**
   b. schnelle Bewegung in die G30-Position erzeugen - **NGC kann das**
6. Senden Sie einen CHANGE_TOOL Canon-Befehl an `task` - * Ausführen in Python Epilog*
7. Einstellen der nummerierten Parameter 5400-5413 entsprechend dem neuen Tool - **Ausführen in Python Epilog**
8. Signal an `task`, den Aufruf des Interpreters für Readahead zu beenden, bis der Werkzeugwechsel

abgeschlossen ist - **im Python-Epilog ausführen**

Wir brauchen also einen Prolog und einen Epilog. Nehmen wir an, unsere INI-Datei Beschwörung der M6-Remap sieht wie folgt aus:

```
REMAP=M6   modalgroup=6  prolog=change_prolog ngc=change epilog=change_epilog
```

Der Prolog, der die Schritte 1 und 2 abdeckt, würde also wie folgt aussehen: Wir beschließen, einige Variablen an die Remap-Prozedur zu übergeben, die dort überprüft und geändert oder in einer Nachricht verwendet werden können. Diese sind: `tool_in_spindle`, `selected_tool` (Werkzeugnummern) und ihre jeweiligen tooldata-Indizes `current_pocket` und `selected_pocket`:

NOTE Die inzwischen nicht mehr verwendeten Namen **selected_pocket** und **current_pocket** verweisen auf einen sequentiellen Werkzeugdatenindex für Werkzeugelemente, die aus einer Werkzeugtabelle ([EMCIO]TOOL_TABLE) oder über eine Werkzeugdatenbank ([EMCIO]DB_PROGRAM) geladen werden.

```
def change_prolog(self, **words):
    try:
        if self.selected_pocket < 0:
            return "M6: no tool prepared"

        if self.cutter_comp_side:
            return "Cannot change tools with cutter radius compensation on"

        self.params["tool_in_spindle"] = self.current_tool
        self.params["selected_tool"] = self.selected_tool
        self.params["current_pocket"] = self.current_pocket
        self.params["selected_pocket"] = self.selected_pocket
        return INTERP_OK
    except Exception as e:
        return "M6/change_prolog: {}".format(e)
```

Sie werden feststellen, dass die meisten Prologfunktionen sehr ähnlich aussehen:

1. Prüfen Sie zunächst, ob alle Voraussetzungen für die Ausführung des Codes erfüllt sind, und dann
2. Vorbereitung der Umgebung - Injektion von Variablen und/oder Durchführung vorbereitender Verarbeitungsschritte, die nicht einfach im NGC-Code durchgeführt werden können;
3. dann an das NGC-Verfahren übergeben, indem Sie INTERP_OK zurücksenden.

Unsere erste Iteration der O-Wort-Prozedur ist unaufregend - wir überprüfen nur, ob wir die Parameter richtig eingegeben haben, und signalisieren den Erfolg, indem wir einen positiven Wert zurückgeben; die Schritte 3-5 werden schließlich hier behandelt (siehe here für die Variablen, die sich auf die Einstellungen der INI-Datei beziehen):

```
O<change> sub
(debug, change: current_tool=#<current_tool>)
(debug, change: selected_pocket=#<selected_pocket>)
;
```

```
; insert any G-code which you see fit here, e.g.:
; G0  #<_ini[setup]tc_x>  #<_ini[setup]tc_y>  #<_ini[setup]tc_z>
;
O<change> endsub [1]
m2
```

Unter der Annahme, dass die "change.ngc" erfolgreich ist, müssen wir die Schritte 6-8 abschließen:

```
def change_epilog(self, **words):
    try:
        if self.return_value > 0.0:
            # commit change
            self.selected_pocket =  int(self.params["selected_pocket"])
            emccanon.CHANGE_TOOL(self.selected_pocket)
            # cause a sync()
            self.tool_change_flag = True
            self.set_tool_parameters()
            return INTERP_OK
        else:
            return "M6 aborted (return code %.1f)" % (self.return_value)

    except Exception, e:
        return "M6/change_epilog: %s" % (e)
```

Dieser Ersatz-M6 ist mit dem eingebauten Code kompatibel, allerdings müssen die Schritte 3-5 mit Ihrem NGC-Code ausgefüllt werden.

Wieder haben die meisten Epiloge ein gemeinsames Schema:

1. Stellen Sie zunächst fest, ob bei der Neukonfiguration alles richtig gelaufen ist,
2. dann alle Übertragungs- und Bereinigungsaktionen durchführen, die nicht im NGC-Code durchgeführt werden können.

Konfigurieren von `iocontrol` mit einem neu zugeordneten M6

Beachten Sie, dass sich die Reihenfolge der Operationen geändert hat: Wir führen alles aus, was in der O-Wort-Prozedur erforderlich ist - einschließlich des Setzens/Lesens von HAL-Pins, um einen Wechsler in Gang zu setzen und einen Werkzeugwechsel zu bestätigen - wahrscheinlich mit `motion.digital-*` und `motion-analog-*` IO-Pins. Wenn wir schließlich den Befehl `CHANGE_TOOL()` ausführen, sind alle Bewegungen und HAL-Interaktionen bereits abgeschlossen.

Normalerweise würde `iocontrol` erst jetzt sein Ding machen, wie in here beschrieben. Wie auch immer, wir brauchen den HAL-Pin nicht mehr - alles, was `iocontrol` noch zu tun hat, ist zu akzeptieren, dass wir mit Prepare fertig sind und wechseln.

Dies bedeutet, dass die entsprechenden `iocontrol`-Pins keine Funktion mehr haben. Daher konfigurieren wir `iocontrol` so, dass es eine Änderung sofort quittiert, indem wir es wie folgt konfigurieren:

```
# loop change signals when remapping M6
```

```
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed
```

Wenn Sie aus irgendeinem Grund Tx neu zuordnen wollen (vorbereiten), müssen die entsprechenden iocontrol-Pins ebenfalls durchgeschleift werden.

Schreiben der Änderung und Vorbereitung der O-Wort-Verfahren

Die Standard-Prologs und Epilogs, die in ncfiles/remap_lib/python-stdglue/stdglue.py zu finden sind, übergeben einige *exponierte Parameter* an die Remap-Prozedur.

Ein "exponierter Parameter" ist eine benannte lokale Variable, die in einer Remap-Prozedur sichtbar ist und einer interpreterinternen Variable entspricht, die für den aktuellen Remap relevant ist. Exponierte Parameter werden im jeweiligen Prolog eingerichtet und im Epilog überprüft. Sie können in der Remap-Prozedur geändert werden und die Änderung wird im Epilog aufgegriffen. Die exponierten Parameter für remappbare Built-in-Codes sind:

- T (prepare_prolog): #<tool> , #<pocket>
- M6 (change_prolog): #<tool_in_spindle>, #<selected_tool>, #<current_pocket>, #<selected_pocket>
- M61 (settool_prolog): #<tool> , #<pocket>
- S (setspeed_prolog): #<speed>
- F (setfeed_prolog): #<feed>

Wenn Sie spezielle Anforderungen an zusätzliche Parameter haben, die sichtbar gemacht werden sollen, können Sie diese einfach zum Prolog hinzufügen - praktisch alle Interpreter-Interna sind für Python sichtbar.

Minimale Änderungen an den eingebauten Codes, einschließlich M6

Denken Sie daran, dass die Neuzuordnung eines Codes normalerweise die gesamte interne Verarbeitung für diesen Code deaktiviert.

In einigen Situationen könnte es jedoch ausreichen, einige Codes um die bestehende eingebaute Implementierung von M6 zu ergänzen, z. B. einen Werkzeuglängentaster, aber ansonsten das Verhalten von M6 beizubehalten.

Da dies ein häufiges Szenario sein kann, wurde das integrierte Verhalten neu zugeordneter Codes innerhalb der Neuzuordnungsprozedur zur Verfügung gestellt. Der Interpreter erkennt, dass Sie sich auf einen neu zugeordneten Code innerhalb der Prozedur beziehen, der sein Verhalten neu definieren soll. In diesem Fall wird das eingebaute Verhalten verwendet - dies ist derzeit für den Satz aktiviert: *M6*, *M61*, *T*, *S*, *F*. Beachten Sie, dass andernfalls das Verweisen auf einen Code innerhalb seiner eigenen Remap-Prozedur ein Fehler wäre - eine "Remapping-Rekursion".

Ein leichtes Verdrehen eines Einbaus würde so aussehen (im Fall von M6):

```
REMAP=M6   modalgroup=6  ngc=mychange
```

```
o<mychange> sub
M6 (verwendet das eingebaute M6-Verhalten)
(... zum Werkzeuglängenschalter fahren, Werkzeuglänge antasten und einstellen...)
o<mychange> endsub
m2
```

CAUTION	Wenn Sie einen eingebauten Code umdefinieren, **geben Sie keine führenden Nullen in G- oder M-Codes** an - sagen Sie zum Beispiel `REMAP=M1 ..`, nicht `REMAP=M01 ...`.

Im Verzeichnis `configs/sim/axis/remap/extend-builtins` finden Sie eine vollständige Konfiguration, die der empfohlene Ausgangspunkt für die eigene Arbeit bei der Erweiterung eingebauter Codes ist.

Angabe des T (vorbereitend)-Ersatzes

Wenn Sie mit der default implementation vertraut sind, müssen Sie dies nicht tun. Aber remapping ist auch ein Weg, um Unzulänglichkeiten in der aktuellen Implementierung zu umgehen, zum Beispiel um nicht zu blockieren, bis der "tool-prepared"-Pin gesetzt ist.

Was Sie zum Beispiel tun könnten, ist: - In einem neu zugeordneten T einfach das Äquivalent des `tool-prepare`-Pins setzen, aber hier **nicht** auf `tool-prepared` warten. - In der entsprechenden neu zugeordneten M6, warten Sie auf das `tool-prepared` ganz am Anfang der O-Wort-Prozedur.

Wieder würden die `iocontrol` Tool-Prepare/Tool-Prepared Pins ungenutzt und ersetzt durch `motion.*` Pins, daher müssten diese Pins in einer Schleife sein:

```
# Schleife zur Vorbereitung von Signalen bei der Neuzuordnung von T
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
```

Hier ist also der Aufbau für ein remapped T:

```
REMAP=T  prolog=prepare_prolog epilog=prepare_epilog ngc=prepare
```

```
def prepare_prolog(self,**words):
    try:
        cblock = self.blocks[self.remap_level]
        if not cblock.t_flag:
            return "T requires a tool number"

        tool  = cblock.t_number
        if tool:
            (status, pocket) = self.find_tool_pocket(tool)
            if status != INTERP_OK:
                return "T%d: pocket not found" % (tool)
        else:
            pocket = -1 # this is a T0 - tool unload

        # these variables will be visible in the ngc O-word sub
        # as #<tool> and #<pocket> local variables, and can be
```

```
        # modified there - the epilog will retrieve the changed
        # values
        self.params["tool"] = tool
        self.params["pocket"] = pocket

        return INTERP_OK
    except Exception, e:
        return "T%d/prepare_prolog: %s" % (int(words['t']), e)
```

Das minimale ngc-Vorbereitungsverfahren sieht wieder so aus:

```
o<prepare> sub
; returning a positive value to commit:
o<prepare> endsub [1]
m2
```

Und der Epilog:

```
def prepare_epilog(self, **words):
    try:
        if self.return_value > 0:
            self.selected_tool = int(self.params["tool"])
            self.selected_pocket = int(self.params["pocket"])
            emccanon.SELECT_TOOL(self.selected_tool)
            return INTERP_OK
        else:
            return "T%d: aborted (return code %.1f)" % (int(self.params["tool"]),self
.return_value)

    except Exception, e:
        return "T%d/prepare_epilog: %s" % (tool,e)
```

Die Funktionen *prepare_prolog* und *prepare_epilog* sind Teil des *standard glue*, der von *nc_files/remap_lib/python-stdglue/stdglue.py* bereitgestellt wird. Dieses Modul ist dazu gedacht, die meisten Standard-Remapping-Situationen auf eine gemeinsame Weise abzudecken.

Fehlerbehandlung: Umgang mit Abbrüchen

Die eingebaute Werkzeugwechselprozedur hat einige Vorkehrungen für den Umgang mit einem Programmabbruch, z.B. Drücken der Escape-Taste in Axis während eines Wechsels. Ihre neu zugewiesene Funktion verfügt über nichts dergleichen, weshalb eine explizite Bereinigung erforderlich sein könnte, wenn ein neu zugewiesener Code abgebrochen wird. Insbesondere kann eine Remap-Prozedur modale Einstellungen festlegen, die nach einem Abbruch nicht mehr aktiv sein sollen. Wenn Ihre Remap-Prozedur beispielsweise Bewegungscodes (G0,G1,G38...) enthält und die Remap-Prozedur abgebrochen wird, bleibt der letzte modale Code aktiv. Sie möchten jedoch höchstwahrscheinlich, dass jede modale Bewegung abgebrochen wird, wenn die Neuzuordnung abgebrochen wird.

Dazu verwenden Sie die Funktion `[RS274NGC]ON_ABORT_COMMAND`. Diese INI-Option spezifiziert einen O-Wort-Prozeduraufruf, der ausgeführt wird, wenn "task" aus irgendeinem Grund die Programmausführung abbricht. `on_abort` empfängt einen einzelnen Parameter, der die Ursache für den Aufruf der Abbruchprozedur angibt, die für eine bedingte Bereinigung verwendet werden könnte.

Die Gründe sind in nml_intf/emc.hh definiert

```
EMC_ABORT_TASK_EXEC_ERROR = 1,
EMC_ABORT_AUX_ESTOP = 2,
EMC_ABORT_MOTION_OR_IO_RCS_ERROR = 3,
EMC_ABORT_TASK_STATE_OFF = 4,
EMC_ABORT_TASK_STATE_ESTOP_RESET = 5,
EMC_ABORT_TASK_STATE_ESTOP = 6,
EMC_ABORT_TASK_STATE_NOT_ON = 7,
EMC_ABORT_TASK_ABORT = 8,
EMC_ABORT_INTERPRETER_ERROR = 9,    // interpreter failed during readahead
EMC_ABORT_INTERPRETER_ERROR_MDI = 10,   // interpreter failed during MDI execution
EMC_ABORT_USER = 100  // user-defined abort codes start here
```

```
[RS274NGC]
ON_ABORT_COMMAND=O <on_abort> call
```

Die vorgeschlagene on_abort-Prozedur würde folgendermaßen aussehen (passen Sie sie an Ihre Bedürfnisse an):

```
o<on_abort> sub

G54 (Nullpunktverschiebungen werden auf den Standardwert gesetzt)
G17 (XY-Ebene auswählen)
G90 (absolut)
G94 (Vorschubmodus: Einheiten/Minute)
M48 (Vorschub- und Geschwindigkeits-Override einstellen)
G40 (Fräserausgleich aus)
M5 (Spindel aus)
G80 (modale Bewegung aufheben)
M9 (Nebel und Kühlmittel aus)

o100 if [#1 eq 5]
    (machine on)
o100 elseif [#1 eq 6]
    (machine off)
o100 elseif [#1 eq 7]
    (estopped)
o100 elseif [#1 eq 8]
    (msg, abort pressed)
o100 else
    (DEBUG, error parameter is [#1])
o100 endif

o<on_abort> endsub
m2
```

CAUTION

Verwenden Sie niemals ein M2 in einem O-Wort-Unterprogramm, auch nicht in diesem. Es wird schwer zu findende Fehler verursachen. Wenn Sie zum Beispiel ein "M2" in einem Unterprogramm verwenden, wird das Unterprogramm nicht ordnungsgemäß beendet und die NGC-Datei des Unterprogramms bleibt offen, nicht Ihr Hauptprogramm.

Stellen Sie sicher, dass sich `on_abort.ngc` im Suchpfad des Interpreters befindet (empfohlener Ort: `SUBROUTINE_PATH`, um Ihr `NC_FILES`-Verzeichnis nicht mit internen Prozeduren zu überladen).

Die Anweisungen in dieser Prozedur stellen in der Regel sicher, dass alle Zustände nach dem Abbruch bereinigt wurden, wie z.B. das ordnungsgemäße Zurücksetzen der HAL-Pins. Ein Beispiel finden Sie unter `configs/sim/axis/remap/rack-toolchange`.

Beachten Sie, dass das Beenden eines remapped Codes durch Rückgabe von INTERP_ERROR aus dem Epilog (siehe vorheriger Abschnitt) auch den Aufruf der Prozedur `on_abort` bewirkt.

Fehlerbehandlung: Fehlschlagen einer NGC-Prozedur mit neu zugeordnetem Code

Wenn Sie in Ihrer Handler-Prozedur feststellen, dass eine Fehlerbedingung aufgetreten ist, verwenden Sie nicht `M2`, um Ihren Handler zu beenden - siehe oben:

Wenn die Anzeige einer Fehlermeldung und das Anhalten des aktuellen Programms ausreichen, verwenden Sie die Funktion `(abort, <message>)`, um den Handler mit einer Fehlermeldung zu beenden. Beachten Sie, dass Sie nummerierte, benannte, INI- und HAL-Parameter im Text wie in diesem Beispiel ersetzen können (siehe auch `tests/interp/abort-hot-comment/test.ngc`):

```
o100 if [..] (some error condition)
     (abort, Bad Things! p42=#42 q=#<q> INI=#<_ini[a]x> pin=#<_hal[component.pin])
o100 endif
```

NOTE Die Erweiterung der INI- und HAL-Variablen ist optional und kann in der Datei INI deaktiviert werden.

Wenn eine feiner abgestufte Wiederherstellungsmaßnahme erforderlich ist, verwenden Sie die im vorherigen Beispiel beschriebene Redewendung:

- Definieren Sie eine Epilog-Funktion, auch wenn es nur darum geht, eine Fehlerbedingung zu signalisieren,
- übergeben Sie einen negativen Wert vom Handler, um den Fehler zu signalisieren,
- überprüfen Sie den Rückgabewert in der Epilog-Funktion,
- ergreifen Sie alle erforderlichen Wiederherstellungsmaßnahmen,
- Geben Sie die Fehlermeldungszeichenfolge aus dem Handler zurück, der die Interpreterfehlermeldung festlegt und das Programm abbricht (ähnlich wie `abort, message=`).

Diese Fehlermeldung wird in der Benutzeroberfläche angezeigt, und wenn INTERP_ERROR zurückgegeben wird, dann wird dieser Fehler wie jeder andere Laufzeitfehler behandelt.

Beachten Sie, dass sowohl `(abort,`__msg__`)` als auch die Rückgabe von INTERP_ERROR aus einem Epilog dazu führt, dass ein ON_ABORT-Handler aufgerufen wird, falls er definiert ist (siehe vorheriger Abschnitt).

9.7.6. Neu-Zuordnung anderer bestehender Codes:

Automatische Gangwahl durch remapping von S (Spindeldrehzahl einstellen)

Ein möglicher Verwendungszweck für einen umprogrammierten S-Code wäre die "automatische Gangwahl" in Abhängigkeit von der Geschwindigkeit. Bei der Neueinstellung würde man prüfen, ob die gewünschte Geschwindigkeit mit der aktuellen Gangeinstellung erreicht werden kann, und andernfalls entsprechend schalten.

Anpassen des Verhaltens von M0, M1

Ein Anwendungsfall für die Neuzuordnung von M0/M1 wäre die Anpassung des Verhaltens des vorhandenen Codes. So könnte es beispielsweise wünschenswert sein, Spindel, Nebel und Flutung während einer M0- oder M1-Programmunterbrechung auszuschalten und diese Einstellungen bei der Wiederaufnahme des Programms wieder einzuschalten.

Für ein vollständiges Beispiel, das genau das tut, siehe `configs/sim/axis/remap/extend-builtins/`, das M1 wie oben beschrieben anpasst.

Anpassen des Verhaltens von M7, M8, M9

Ein Beispiel für das Remapping des eingebauten Verhaltens von M7/M8/M9 ist die Möglichkeit, optionale Argumente wie ein P-Wort zu übergeben, um eine komplexere Kühlmittelsteuerung zu ermöglichen (z. B. über Werkzeug- versus externe Kühlmittelzufuhr).

Siehe `configs/sim/axis/remap/extend-builtins/` für ein Beispiel einer solchen Erweiterung des eingebauten Verhaltens von M7, M8 und M9.

9.7.7. Neue G-Code-Zyklen erstellen

Ein G-Code-Zyklus, wie hier verwendet, soll sich wie folgt verhalten:

- Beim ersten Aufruf werden die zugehörigen Wörter gesammelt und der G-Code-Zyklus wird ausgeführt.
- Wenn nachfolgende Zeilen nur Parameterwörter fortführen, die für diesen Code gelten, aber keinen neuen G-Code, wird der vorherige G-Code mit entsprechend geänderten Parametern erneut ausgeführt.

Ein Beispiel: Angenommen, Sie haben `G84.3` als remapped G-code cycle mit dem folgenden INI-Segment definiert (siehe here für eine detaillierte Beschreibung von `cycle_prolog` und `cycle_epilog`):

```
[RS274NGC]
# Ein Zyklus mit einem O-Wort-Verfahren: G84.3 <X- Y- Z- Q- P->
REMAP=G84.3 argspec=xyzabcuvwpr prolog=cycle_prolog ngc=g843 epilog=cycle_epilog
modalgroup=1
```

Ausführen der folgenden Zeilen:

```
g17
(1)   g84.3 x1 y2 z3  r1
(2)   x3 y4 p2
(3)   x6 y7 z5
(4)   G80
```

bewirkt Folgendes (beachten Sie, dass "R" klebrig ist und "Z" klebrig ist, da die Ebene "XY" ist):

1. `g843.ngc` wird mit den Worten x=1, y=2, z=3, r=1 aufgerufen
2. `g843.ngc` wird mit den Worten x=3, y=4, z=3, p=2, r=1 aufgerufen
3. `g843.ngc` wird mit den Worten x=6, y=7, z=3, r=1 aufgerufen
4. Der `G84.3`-Zyklus wird abgebrochen.

Neben der Erstellung neuer Zyklen bietet dies eine einfache Methode, um bestehende G-Codes, die sich nicht als Zyklen verhalten, neu zu verpacken. Zum Beispiel verhält sich der Code "G33.1" (Rigid Tapping) nicht wie ein Zyklus. Mit einem solchen Wrapper kann leicht ein neuer Code erstellt werden, der `G33.1` verwendet, sich aber wie ein Zyklus verhält.

Unter "configs/sim/axis/remap/cycle" finden Sie ein vollständiges Beispiel für diese Funktion. Es enthält zwei Zyklen, einen mit einer NGC-Prozedur wie oben, und ein Zyklusbeispiel, das nur Python verwendet.

9.7.8. Embedded Python konfigurieren

Das Python-Plugin dient sowohl als Interpreter als auch als `Task`, wenn es so konfiguriert ist, und hat daher seinen eigenen Abschnitt `PYTHON` in der INI-Datei.

Python plugin : INI-Datei-Konfiguration

`[PYTHON]`

`TOPLEVEL =` <***Dateiname***>

Dateiname des anfänglichen Python-Skripts, das beim Starten ausgeführt wird. Dieses Skript ist für die Einrichtung der Paketnamensstruktur verantwortlich, siehe unten.

`PATH_PREPEND =` <***Verzeichnis***>

Dieses Verzeichnis dem `PYTHON_PATH` voranstellen. Eine sich wiederholende Gruppe.

`PATH_APPEND =` <***Verzeichnis***>

Dieses Verzeichnis an `PYTHON_PATH` anhängen. Eine sich wiederholende Gruppe.

`LOG_LEVEL =` <***Ganzzahl***>

Protokollierungsstufe für Plugin-bezogene Aktionen. Erhöhen Sie diesen Wert, wenn Sie Probleme vermuten. Kann sehr ausführlich sein.

`RELOAD_ON_CHANGE =` **[0|1]**

Lädt das *TOPLEVEL*-Skript neu, wenn die Datei geändert wurde. Praktisch für die Fehlersuche, verursacht aber derzeit einen gewissen Laufzeit-Overhead. Schalten Sie dies für Produktionskonfigurationen aus.

Ausführen von Python-Anweisungen vom Interpreter

Für die Ad-hoc-Ausführung von Befehlen wurde der Python *hot comment* hinzugefügt. Die Python-Ausgabe geht standardmäßig nach stdout, so dass Sie LinuxCNC aus einem Terminal-Fenster starten müssen, um die Ergebnisse zu sehen, zum Beispiel im MDI-Fenster:

```
;py,print(2*3)
```

Beachten Sie, dass die Interpreterinstanz hier als `this` zur Verfügung steht, so dass Sie sie auch ausführen könnten:

```
;py,print(this.tool_table[0].toolno)
```

Hier ist ein Ansatz, um eine O-Word-Unterroutine zu verwenden, die einen Eintrag aus einer Einstellungsdatei ausliest und als G-Code-Parameter hinzufügt.

```
(filename myofile.ngc)
o<myofile> sub

;py,from interpreter import *
;py,import os
;py,from qtvcp.lib.preferences import Access

; Finde und gebe aus den Pfad für die Einstellungs-Datei
;py,CONFPATH = os.environ.get('CONFIG_DIR', '/dev/null')
; Nimm eine Anpassung für den Namen Deiner Einstellungsdatei vor
;py,PREFFILE = os.path.join(CONFPATH,'qtdragon.pref')
;py,print(PREFFILE)

; Erhalte Instanz für Einstellungen
;py,Pref = Access(PREFFILE)

; Lade eine Einstellung und gebe sie aus
;py,this.params['toolToLoad']=Pref.getpref('Tool to load', 0, int,'CUSTOM_FORM_ENTRIES')
;py,print('Tool to load->:',this.params['toolToLoad'])

; Liefere den Wert zurück
o<myofile> endsub [#<toolToLoad>]
M2
```

9.7.9. Programmierung von Embedded Python im RS274NGC Interpreter

Der Python-Plugin-Namensraum

Es wird erwartet, dass der Namespace wie folgt angelegt wird:

`oword`

Alle Aufrufe in diesem Modul sind Kandidaten für Python O-Word-Prozeduren. Beachten Sie, dass das Python `oword` Modul **vor** dem Testen auf eine NGC-Prozedur mit dem gleichen Namen überprüft wird - es werden Namen in `oword` gegenüber NGC-Dateien mit demselben Basisnamen priorisiert.

remap

Die in einer argspec *prolog*, *epilog* oder *python* Option referenzierten Python-Callables werden hier erwartet.

namedparams

Die Python-Funktionen in diesem Modul erweitern oder definieren den Namensraum der vordefinierten benannten Parameter, siehe adding predefined parameters.

Der Interpreter aus der Sicht von Python

Der Interpreter ist eine bestehende C++-Klasse ("Interp"), die in "src/emc/rs274ngc" definiert ist. Konzeptionell sind alle `oword.<function>` und `remap.<function>` Python-Aufrufe Methoden dieser Interp-Klasse, obwohl es keine explizite Python-Definition dieser Klasse gibt (es handelt sich um eine *Boost.Python*-Wrapper-Instanz) und daher den ersten Parameter `self` erhalten, der für den Zugriff auf Interna verwendet werden kann.

Die Interpreterfunktionen `__init__` und `__delete__`

Wenn das Modul `TOPLEVEL` eine Funktion `__init__` definiert, wird diese aufgerufen, sobald der Interpreter vollständig konfiguriert ist (INI-Datei gelesen und Zustand mit dem Weltmodell synchronisiert).

Wenn das Modul `TOPLEVEL` eine Funktion `__delete__` definiert, wird sie einmal aufgerufen, bevor der Interpreter heruntergefahren wird und nachdem die persistenten Parameter in der `PARAMETER_FILE` gespeichert worden sind.

Hinweis_ Zur Zeit funktioniert der `__delete__`-Handler nicht für Interpreter-Instanzen, die durch den Import des `gcode`-Moduls erzeugt wurden. Wenn Sie dort eine gleichwertige Funktionalität benötigen (was ziemlich unwahrscheinlich ist), ziehen Sie bitte das Python-Modul `atexit` in Betracht.

```
# Dies würde im Modul TOPLEVEL definiert werden.

def __init__(self):
    # fügen Sie hier eine einmalige Initialisierung hinzu
    if self.task:
  # dies ist die Milltask-Instanz von interp
  pass
    else:
  # dies ist eine Nicht-Milltask-Instanz von interp
        pass

def __delete__(self):
    # hier alle Aufräum-/Zustandssicherungsaktionen hinzufügen
    if self.task: # wie oben
  pass
    else:
        pass
```

Diese Funktion kann verwendet werden, um alle Python-seitigen Attribute zu initialisieren, die später benötigt werden könnten, z.B. in remap- oder O-word-Funktionen, und um einen Zustand zu speichern

oder wiederherzustellen, der über das hinausgeht, was PARAMETER_FILE bietet.

Wenn es Einrichtungs- oder Aufräumaktionen gibt, die nur in der milltask-Interpreter-Instanz stattfinden sollen (im Gegensatz zu der Interpreter-Instanz, die im gcode-Python-Modul sitzt und der Vorschau/Fortschrittsanzeige dient, aber sonst nichts), kann dies durch evaluating *self.task* getestet werden.

Ein Beispiel für die Verwendung von __init__ und __delete__ findet sich in configs/sim/axis/remap/cycle/python/toplevel.py, das Attribute initialisiert, die für die Handhabung von Zyklen in ncfiles/remap_lib/python-stdglue/stdglue.py benötigt werden (und in configs/sim/axis/remap/cycle/python/remap.py importiert wurde).

Aufrufkonventionen: NGC zu Python

Python-Code wird in den folgenden Situationen von NGC aufgerufen:

- bei normaler programmausführung:
  - wenn ein O-Wort-Aufruf wie O<proc> call ausgeführt wird und der Name oword.proc definiert und aufrufbar ist
  - wenn ein Kommentar wie `;py,<Python statement>`ausgeführt wird.
  - während der Ausführung eines umgewandelten Codes: alle prolog=, python= und epilog= Handler.

Aufruf von O-Wort-Python-Unterroutinen

Argumente:

self

Die Interpreter-Instanz.

***args**

Die Liste der tatsächlichen Positionsparameter. Da die Anzahl der aktuellen Parameter variieren kann, ist es am besten, diese Art der Deklaration zu verwenden:

```
#  dies würde im oword-Modul definiert werden
def mysub(self, *args):
    print("number of parameters passed:", len(args))
    for a in args:
        print(a)
```

Rückgabewerte von O-Wort-Python-Unterroutinen

Genauso wie NGC-Prozeduren Werte zurückgeben können, tun dies auch O-Word-Python-Unterprogramme. Von ihnen wird erwartet, dass sie entweder zurückgeben

- keinen Wert (keine "Return"-Anweisung oder der Wert "None"),
- ein Gleitkomma- (engl. float) oder Ganzzahl (engl. int(eger))-Wert,
- eine Zeichenkette, etwa *Dies ist eine Fehlermeldung, brechen Sie das Programm ab*. Funktioniert wie

`(abort, msg)`.

Jeder andere Rückgabewerttyp löst eine Python-Ausnahme aus.

In einer aufrufenden NGC-Umgebung sind die folgenden vordefinierten benannten Parameter verfügbar:

#<value>

Wert, der von der zuletzt aufgerufenen Prozedur zurückgegeben wurde. Beim Start auf 0.0 initialisiert. Wird in Interp als `self.return_value` (float) angezeigt.

#<value_returned>

Zeigt an, dass die zuletzt aufgerufene Prozedur `return` oder `endsub` mit einem expliziten Wert zurückgegeben hat. 1.0 wenn wahr. Wird bei jedem `Aufruf` auf 0.0 gesetzt. Ausgesetzt in Interp war `self.value_returned` (int).

Siehe auch `tests/interp/value-returned` für ein Beispiel.

Aufrufkonventionen für prolog=- und epilog=-Unterroutinen

Argumente sind:

self

Die Interpreter-Instanz.

words

Schlüsselwort-Parameter-Wörterbuch. Wenn ein argspec vorhanden war, werden die Wörter entsprechend aus dem aktuellen Block gesammelt und der Einfachheit halber an das Wörterbuch übergeben (die Wörter könnten auch direkt aus dem aufrufenden Block geholt werden, aber das erfordert mehr Wissen über die Interpreter-Interna). Wenn kein argspec übergeben wurde oder nur optionale Werte angegeben wurden und keiner dieser Werte im aufrufenden Block vorhanden war, ist dieses `dict` leer. Wortnamen werden in Kleinbuchstaben umgewandelt.

Beispielaufruf:

```
def minimal_prolog(self, **words): # in remap module
    print(len(words)," words passed")
    for w in words:
        print("%s: %s" % (w, words[w]))
    if words['p'] < 78: # NB: could raise an exception if p were optional
       return "failing miserably"
    return INTERP_OK
```

Rückgabewerte:

INTERP_OK

Gibt dies bei Erfolg zurück. Sie müssen dies aus "Interpreter" importieren.

Text einer Nachricht

Die Rückgabe einer Zeichenkette von einem Handler bedeutet *dies ist eine Fehlermeldung, breche das*

Programm ab. Funktioniert wie `(abort, msg )`.

Aufrufkonventionen für python=-Unterroutinen

Argumente sind:

`self`

Die Interpreter-Instanz.

`words`

Schlüsselwort-Parameter-Wörterbuch. Dasselbe kwargs-Wörterbuch wie Prologs und Epilogs (siehe oben).

Das minimale *python*=-Funktionsbeispiel:

```
def useless(self,  **words): # in remap module
    return INTERP_OK
```

Rückgabewerte:

`INTERP_OK`

Bei Erfolg wird dies zurückgegeben

Text einer Nachricht

Die Rückgabe einer Zeichenkette von einem Handler bedeutet *dies ist eine Fehlermeldung, breche das Programm ab*. Funktioniert wie `(abort, msg )`.

Wenn der Handler eine "Queuebuster-Operation" (Werkzeugwechsel, Messtaster, HAL-Pin-Lesen) ausführen muss, dann soll er die Ausführung mit der folgenden Anweisung unterbrechen:

`yield INTERP_EXECUTE_FINISH`

Dies signalisiert `task`, das Weiterlesen zu stoppen, alle Operationen in der Warteschlange auszuführen, die Operation "queue-buster" auszuführen, den Zustand des Interpreters mit dem Zustand der Maschine zu synchronisieren und dann dem Interpreter zu signalisieren, fortzufahren. An diesem Punkt wird die Funktion an der Anweisung nach der Anweisung "yield ..`" fortgesetzt.

Umgang mit Queue-Buster: Sonde, Werkzeugwechsel und Warten auf einen HAL-Pin

Queue-Buster unterbrechen eine Prozedur an dem Punkt, an dem eine solche Operation aufgerufen wird, so dass die Prozedur nach dem Interpreter synch() neu gestartet werden muss. Wenn dies geschieht, muss die Prozedur wissen, ob sie neu gestartet wurde und wo sie fortfahren soll. Die Python-Generator-Methode wird verwendet, um den Neustart einer Prozedur zu bewältigen.

Dies zeigt die Fortsetzung des Anrufs mit einem einzigen Ausgangspunkt:

```
def read_pin(self,*args):
    # 5 Sekunden warten, bis Digital-Eingang 00 auf High geht
    emccanon.WAIT(0,1,2,5.0)
    # übergebe die Kontrolle nach der Ausführung des Queue Busters:
    yield INTERP_EXECUTE_FINISH
    # Post-sync()-Ausführung wird hier fortgesetzt:
```

```
    pin_status = emccanon.GET_EXTERNAL_DIGITAL_INPUT(0,0);
    print("pin status=",pin_status)
```

WARNING Die Funktion *yield* ist anfällig. Die folgenden Einschränkungen gelten für die Verwendung von *yield INTERP_EXECUTE_FINISH*:

- Python-Code, der ein *yield INTERP_EXECUTE_FINISH* ausführt, muss Teil einer Remap-Prozedur sein. Yield funktioniert nicht in einer Python-O-word-Prozedur.
- Eine Python-Remap-Subroutine, welche die Anweisung *yield INTERP_EXECUTE_FINISH* enthält, darf keinen Wert zurückgeben, wie dies bei normalen Python-Yield-Anweisungen der Fall ist.
- Code, der einem Yield folgt, darf den Interpreter nicht rekursiv aufrufen, wie bei `self.execute("<mdi command>")`. Dies ist eine architektonische Einschränkung des Interpreters und kann nicht ohne ein größeres Redesign behoben werden.

Aufrufkonventionen: Python zu NGC

NGC-Code wird von Python ausgeführt, wenn

- die Methode `self.execute(<NGC-Code>[,<Zeilennummer>])` ausgeführt wird, oder
- Wenn bei der Ausführung eines neu zugeordneten Codes eine `prolog=`-Funktion definiert ist, wird die in `ngc=` angegebene NGC-Prozedur unmittelbar danach ausgeführt.

Der Prolog-Handler ruft den Handler nicht auf, sondern bereitet dessen Aufrufumgebung vor, indem er z. B. vordefinierte lokale Parameter einrichtet.

Einfügen von Parametern in einen Prolog und Abrufen von Parametern in einem Epilog

Konzeptionell werden ein Prolog und ein Epilog auf der gleichen Aufrufebene wie die O-Wort-Prozedur ausgeführt, d. h. nach dem Aufbau des Unterprogrammaufrufs und vor dem Ende des Unterprogramms oder der Rückkehr.

Das bedeutet, dass jede lokale Variable, die in einem Prolog erstellt wird, eine lokale Variable in der O-Wort-Prozedur ist, und dass alle lokalen Variablen, die in der O-Wort-Prozedur erstellt werden, immer noch zugänglich sind, wenn der Epilog ausgeführt wird.

Das Array "self.params" dient zum Lesen und Setzen von nummerierten und benannten Parametern. Wenn ein benannter Parameter mit _ (Unterstrich) beginnt, wird angenommen, dass er ein globaler Parameter ist; wenn nicht, ist er lokal für die aufrufende Prozedur. Auch nummerierte Parameter im Bereich 1..30 werden wie lokale Variablen behandelt; ihre ursprünglichen Werte werden bei Return/Endsub einer O-Wort-Prozedur wiederhergestellt.

Hier ist ein Beispiel für umgewandelten Code, der das Einfügen und Extrahieren von Parametern in/aus der O-Wort-Prozedur demonstriert:

```
REMAP=m300 prolog=insert_param ngc=testparam epilog=retrieve_param modalgroup=10
```

```
def insert_param(self, **words): # in the remap module
    print("insert_param call level=",self.call_level)
```

```
    self.params["myname"] = 123
    self.params[1] = 345
    self.params[2] = 678
    return INTERP_OK

def retrieve_param(self, **words):
    print("retrieve_param call level=",self.call_level)
    print("#1=", self.params[1])
    print("#2=", self.params[2])
    try:
        print("result=", self.params["result"])
    except Exception,e:
  return "testparam forgot to assign #<result>"
    return INTERP_OK
```

```
o<testparam> sub
(debug, call_level=#<_call_level> myname=#<myname>)
; try commenting out the next line and run again
#<result> = [#<myname> * 3]
#1 = [#1 * 5]
#2 = [#2 * 3]
o<testparam> endsub
m2
```

Die Funktion `self.params()` gibt eine Liste aller derzeit definierten Variablennamen zurück. Da `myname` lokal ist, verschwindet es nach Beendigung des Epilogs.

Aufruf des Interpreters aus Python

Sie können den Interpreter aus dem Python-Code wie folgt rekursiv aufrufen:

```
self.execute(<NGC code>[,<line number>])
```

Beispiele:

```
  self.execute("G1 X%f Y%f" % (x,y))
  self.execute("O <myprocedure> call", currentline)
```

Sie sollten prüfen, ob der Rückgabewert `< INTERP_MIN_ERROR` ist. Wenn Sie viele execute()-Anweisungen verwenden, ist es wahrscheinlich einfacher, eine InterpreterException wie unten beschrieben abzufangen.

CAUTION

Die im vorherigen Abschnitt beschriebene Methode zum Einfügen/Abrufen von Parametern funktioniert in diesem Fall nicht. Sie ist gut genug für

- die Ausführung einfacher NGC-Befehle oder eines Prozeduraufrufs und
- fortgeschrittene Selbstbeobachtung (engl. introspektion) des Verfahrens und
- Die Übergabe von lokalen benannten Parametern ist nicht erforderlich.

Die Funktion des rekursiven Aufrufs ist anfällig für Störungen.

Interpreter-Ausnahme während execute()

wenn `interpreter.throw_exceptions` ungleich Null ist (Standardwert 1) und self.execute() einen Fehler zurückgibt, wird die Ausnahme `InterpreterException` ausgelöst. InterpreterException hat die folgenden Attribute:

`Zeilennummer`

wo der Fehler aufgetreten ist

`zeilen_text`

die NGC-Anweisung, die den Fehler verursacht

`Fehlermeldung`

die Fehlermeldung des Interpreters

Fehler können auf die folgende Python-Weise abgefangen werden:

```
import interpreter
interpreter.throw_exceptions = 1
   ...
   try:
        self.execute("G3456")  #  raise InterpreterException

   except InterpreterException,e:
        msg = "%d: '%s' - %s" % (e.line_number,e.line_text, e.error_message)
        return msg  # ersetzt regulär ausgegebene Fehlermeldung
```

Canon

Die Kanonebene besteht praktisch nur aus freien Funktionen. Beispiel:

```
import emccanon
def example(self,*args):
    ....
    emccanon.STRAIGHT_TRAVERSE(line,x0,y0,z0,0,0,0,0,0,0)
    emccanon.STRAIGHT_FEED(line,x1,y1,z1,0,0,0,0,0,0)
    ...
    return INTERP_OK
```

Die eigentlichen Kanon-Funktionen sind in `src/emc/nml_intf/canon.hh` deklariert und in `src/emc/task/emccanon.cc` implementiert. Die Implementierung der Python-Funktionen ist in `src/emc/rs274ncg/canonmodule.cc` zu finden.

Eingebaute Module

Die folgenden Module sind bereits integriert:

`interpreter`

Legt Interna der Interp-Klasse offen. Siehe `src/emc/rs274ngc/interpmodule.cc`, und den `tests/remap/introspect` Regressionstest.

emccanon

Legt die meisten Aufrufe von `src/emc/task/emccanon.cc` offen.

9.7.10. Hinzufügen vordefinierter benannter Parameter

Der Interpreter verfügt über eine Reihe von vordefinierten benannten Parametern für den Zugriff auf den internen Status auf NGC-Sprachebene. Diese Parameter sind schreibgeschützt und global und können daher nicht zugewiesen werden.

Zusätzliche Parameter können durch die Definition einer Funktion im Modul `namedparams` hinzugefügt werden. Der Name der Funktion definiert den Namen des neuen vordefinierten benannten Parameters, der nun in beliebigen Ausdrücken referenziert werden kann.

Um einen benannten Parameter hinzuzufügen oder neu zu definieren:

- Ein Modul `namedparams` hinzufügen, damit es vom Interpreter gefunden werden kann,
- neue Parameter durch Funktionen definieren (siehe unten). Diese Funktionen erhalten `self` (die Interpreterinstanz) als Parameter und können so auf beliebige Zustände zugreifen. Beliebige Python-Fähigkeiten können verwendet werden, um einen Wert zurückzugeben.
- Importieren Sie dieses Modul aus dem `TOPLEVEL`-Skript.

```
# namedparams.py
# trivial example
def _pi(self):
    return 3.1415926535
```

```
#<Umfang> = [2 * #<Radius> * #<_pi>]
```

Von den Funktionen in `namedparams.py` wird erwartet, dass sie einen float- oder int-Wert zurückgeben. Wenn ein String zurückgegeben wird, dann wird eine Fehlermeldung des Interpreters gesetzt und die Ausführung abgebrochen.

Es werden nur Funktionen mit führendem Unterstrich als Parameter hinzugefügt, da dies die RS274NGC-Konvention für Globals ist.

Es ist möglich, einen vorhandenen vordefinierten Parameter umzudefinieren, indem eine Python-Funktion gleichen Namens zum Modul `namedparams` hinzugefügt wird. In diesem Fall wird beim Starten eine Warnung ausgegeben.

Das obige Beispiel ist zwar nicht sonderlich nützlich, aber beachten Sie, dass so ziemlich der gesamte interne Zustand des Interpreters von Python aus zugänglich ist, so dass beliebige Prädikate auf diese Weise definiert werden können. Für ein etwas fortgeschritteneres Beispiel, siehe `tests/remap/predefined-named-params`.

9.7.11. Standardmäßige Glue (Programmierer-Slang für verbindende)-Routinen

Da viele Mapping-Aufgaben sehr ähnlich sind, habe ich begonnen, funktionierende Prolog- und Epilog-

Routinen in einem einzigen Python-Modul zu sammeln. Diese sind derzeit in *ncfiles/remap_lib/python-stdglue/stdglue.py* zu finden und bieten die folgenden Routinen:

T: `prepare_prolog` und `prepare_epilog`

Diese verpacken ein NGC-Verfahren für T*x* Tool Prepare.

Aktionen von `prepare_prolog`

Die folgenden Parameter werden für das NGC-Verfahren sichtbar gemacht:

- `#<tool>` - der Parameter des `T`-Wortes
- `#<pocket>` - die entsprechende Tasche

Wenn die Werkzeugnummer Null angefordert wird (d.h. Werkzeug entladen), wird die entsprechende Tasche als -1 übergeben.

Es ist ein Fehler, wenn:

- Keine Werkzeugnummer als T-Parameter angegeben ist,
- das Werkzeug nicht in der Werkzeugtabelle gefunden werden kann.

Beachten Sie, dass Werkzeug und Platznummer identisch sind und die Platznummer aus der Werkzeugtabelle ignoriert wird, wenn Sie nicht den Parameter `[EMCIO] RANDOM_TOOLCHANGER=1` setzen. Dies ist derzeit eine Einschränkung.

Aktionen von `prepare_epilog`

- Von der NGC-Prozedur wird erwartet, dass sie einen positiven Wert zurückgibt, andernfalls wird eine Fehlermeldung mit dem Rückgabewert ausgegeben und der Interpreter bricht ab.
- Wenn die NGC-Prozedur den T-Befehl ausführt (der sich dann auf das eingebaute T-Verhalten bezieht), wird keine weitere Aktion ausgeführt. Dies kann z. B. genutzt werden, um das eingebaute Verhalten minimal anzupassen, indem man ihm einige andere Anweisungen voran- oder nachstellt.
- Andernfalls werden die Parameter `#<tool>` und `#<pocket>` aus dem Parameterraum des Unterprogramms extrahiert. Das bedeutet, dass die NGC-Prozedur diese Werte ändern könnte, und der Epilog berücksichtigt die geänderten Werte.
- Dann wird der Canon-Befehl `SELECT_TOOL(#<tool>)` ausgeführt.

M6: `change_prolog` und `change_epilog`

Diese schließen ein NGC-Verfahren für den M6-Werkzeugwechsel ein.

Aktionen von `change_prolog`

- Wenn es keinen vorhergehenden T-Befehl gab, der die Auswahl einer Tasche zur Folge hatte, bricht der Prolog mit einer Fehlermeldung ab.
- Wenn die Fräserradiuskompensation eingeschaltet ist, bricht der Prolog mit einer Fehlermeldung ab.

Anschließend werden die folgenden Parameter in das NGC-Verfahren exportiert:

- #<tool_in_spindle> : die Werkzeugnummer des aktuell geladenen Werkzeugs
- #<selected_tool> : die Nummer des ausgewählten Werkzeugs
- #<selected_pocket> : der Index der Werkzeugdaten des ausgewählten Werkzeugs

Aktionen von +change_epilog

- Von der NGC-Prozedur wird erwartet, dass sie einen positiven Wert zurückgibt, andernfalls wird eine Fehlermeldung mit dem Rückgabewert ausgegeben und der Interpreter bricht ab.
- Wenn die NGC-Prozedur den M6-Befehl ausführt (der sich dann auf das eingebaute M6-Verhalten bezieht), wird keine weitere Aktion ausgeführt. Dies kann z. B. genutzt werden, um das eingebaute Verhalten minimal anzupassen, indem man ihm einige andere Anweisungen voran- oder nachstellt.
- Andernfalls wird der Parameter #<selected_pocket> aus dem Parameterraum des Unterprogramms extrahiert und verwendet, um die Variable current_pocket des Interpreters zu setzen. Auch hier kann die Prozedur diesen Wert ändern, und der Epilog berücksichtigt den geänderten Wert.
- Dann wird der Canon-Befehl CHANGE_TOOL(#<selected_pocket>) ausgeführt.
- Die neuen Werkzeugparameter (Versatz, Durchmesser usw.) werden eingestellt.

G-Code-Zyklen: cycle_prolog und cycle_epilog

Diese umhüllen eine NGC-Prozedur, so dass sie als Zyklus fungieren kann, was bedeutet, dass der Bewegungscode nach Abschluss der Ausführung erhalten bleibt. Wenn die nächste Zeile nur Parameterwörter enthält (z. B. neue X- und Y-Werte), wird der Code erneut ausgeführt, wobei die neuen Parameterwörter in die Menge der beim ersten Aufruf angegebenen Parameter eingefügt werden.

Diese Routinen sind so konzipiert, dass sie in Verbindung mit einem argspec=<words>-Parameter arbeiten. Dies ist zwar einfach zu verwenden, aber in einem realistischen Szenario würden Sie argspec vermeiden und den Block manuell gründlicher untersuchen, um bessere Fehlermeldungen zu erhalten.

Der Vorschlag für argspec lautet wie folgt:

```
REMAP=G<somecode> argspec=xyzabcuvwqplr prolog=cycle_prolog ngc=<ngc procedure>
epilog=cycle_epilog modalgroup=1
```

Auf diese Weise kann cycle_prolog die Kompatibilität der im Block angegebenen Achsenwörter ermitteln (siehe unten).

Aktionen von cycle_prolog

- Ermitteln Sie, ob die vom aktuellen Block übergebenen Wörter die unter Canned Cycle Errors genannten Bedingungen erfüllen.
  - Exportiert die Achsenwörter als <x>, #<y> usw.; schlägt fehl, wenn Achsenwörter aus verschiedenen Gruppen (XYZ) (UVW) zusammen verwendet werden oder eines von (ABC) angegeben wird.
  - Exportiere *L*- als #<l>; Standardwert ist 1, wenn nicht angegeben.
  - Exportiere *P*- als #<p>; scheitere, wenn p kleiner als 0.

- Exportiere *R*- als `#<r>`; scheitert, wenn r nicht gegeben ist, oder kleiner gleich 0, wenn gegeben.
- Fehler, wenn die Vorschubrate null ist oder der inverse Zeitvorschub oder die Fräserkompensation eingeschaltet ist.

- Feststellen, ob dies der erste Aufruf eines Zyklus-G-Codes ist, falls ja:
  - Fügen Sie die (gemäß argspec) übergebenen Wörter zu einem Satz von Sticky-Parametern hinzu, der über mehrere Aufrufe hinweg beibehalten wird.
- Wenn nicht (eine Fortsetzungszeile mit neuen Parametern), dann
  - die übergebenen Wörter in den bestehenden Satz von Sticky-Parametern einfügen.
- Exportieren Sie den Satz der Sticky-Parameter in das NGC-Verfahren.

Aktionen von `cycle_epilog`

- Feststellen, ob der aktuelle Code tatsächlich ein Zyklus war, wenn ja, dann
  - den aktuellen Bewegungsmodus beibehalten, so dass eine Fortsetzungszeile ohne Bewegungscode denselben Bewegungscode ausführt.

S (Geschwindigkeit einstellen) : `setspeed_prolog` und `setspeed_epilog`

TBD

F (Vorschub einstellen) : `setfeed_prolog` und `setfeed_epilog`

TBD

M61 Werkzeugnummer einstellen: `settool_prolog` und `settool_epilog`

TBD

9.7.12. Remaped Code Ausführung

NGC-Prozeduraufrufumgebung bei Remaps

Normalerweise wird eine O-Wort-Prozedur mit Positionsparametern aufgerufen. Dieses Schema ist sehr einschränkend, insbesondere wenn optionale Parameter vorhanden sind. Daher wurde die Aufrufkonvention erweitert, um etwas zu verwenden, das dem Python-Modell für Schlüsselwortargumente sehr ähnlich ist.

Siehe LINKTO G-Code/Main Subroutinen: sub, endsub, return, call.

Verschachtelte remapped Codes

Neu zugeordnete Codes können wie Prozeduraufrufe verschachtelt werden, d. h. ein neu zugeordneter Code, dessen NGC-Prozedur auf einen anderen neu zugeordneten Code verweist, wird korrekt ausgeführt.

Die maximale Verschachtelungsebene, die neu zugeordnet werden kann, beträgt derzeit 10.

Laufende Nummer bei Remaps

Sequenznummern werden wie bei O-Wort-Aufrufen propagiert und wiederhergestellt. Siehe `tests/remap/nested-remaps/word` für den Regressionstest, der die Verfolgung der Sequenznummer bei verschachtelten Remaps drei Ebenen tief zeigt.

Debugging-Flags

Die folgenden Flags sind für das Mapping und die Ausführung in Python relevant:

`EMC_DEBUG_OWORD`	`0x00002000`	verfolgt die Ausführung von O-Wort-Unterprogrammen
`EMC_DEBUG_REMAP`	`0x00004000`	verfolgt die Ausführung von remap-bezogenem Code
`EMC_DEBUG_PYTHON`	`0x00008000`	Aufrufe für das Python-Plugin
`EMC_DEBUG_NAMEDPARAM`	`0x00010000`	Zugriff auf benannte Parameter verfolgen
`EMC_DEBUG_USER1`	`0x10000000`	Benutzerdefiniert - nicht von LinuxCNC interpretiert
`EMC_DEBUG_USER2`	`0x20000000`	Benutzerdefiniert - nicht von LinuxCNC interpretiert

oder diese Flags in die '[EMC]DEBUG'-Variable nach Bedarf. Eine aktuelle Liste der Debug-Flags finden Sie in *src/emc/nml_intf/debugflags.h*.

Fehlersuche in eingebettetem Python-Code

Das Debuggen von eingebettetem Python-Code ist schwieriger als das Debuggen von normalen Python-Skripten, und es gibt nur ein begrenztes Angebot an Debuggern. Eine funktionierende Lösung auf Open-Source-Basis ist die Verwendung der Eclipse IDE und des PydDev Eclipse Plug-ins und seiner Remote-Debugging-Funktion.

Um diesen Ansatz zu verwenden:

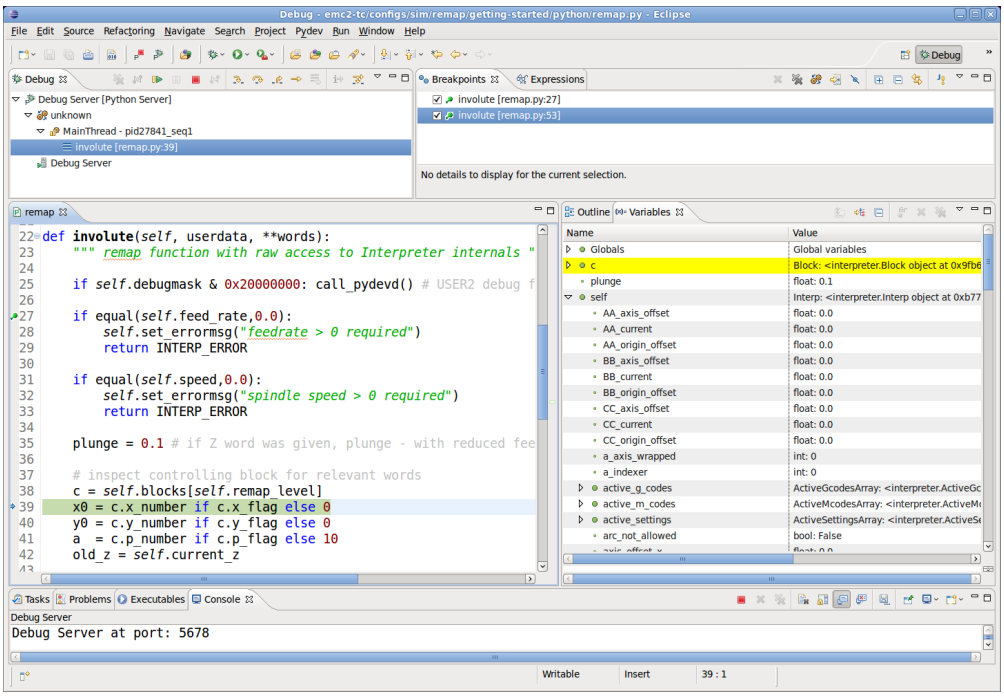
- Installieren Sie Eclipse über das *Ubuntu Software Center* (wählen Sie die erste Option).
- Installieren Sie das PyDev-Plug-in von der Pydev Update Site.
- Richten Sie den LinuxCNC-Quellbaum als Eclipse-Projekt ein.
- Starten Sie den Pydev Debug Server in Eclipse.
- Stellen Sie sicher, dass der eingebettete Python-Code das Modul `pydevd.py` finden kann, das mit diesem Plugin geliefert wird - es ist irgendwo tief im Eclipse-Installationsverzeichnis vergraben. Setzen Sie die Variable `pydevd` in der Datei `util.py` so, dass sie dieses Verzeichnis wiedergibt.

- Fügen Sie `import pydevd` zu Ihrem Python-Modul hinzu - siehe Beispiel `util.py` und `remap.py`.
- Rufen Sie irgendwann `pydevd.settrace()` in Ihrem Modul auf, um sich mit dem Eclipse Python Debug Server zu verbinden - hier können Sie wie gewohnt Haltepunkte in Ihrem Code setzen, Variablen inspizieren, Schritte machen usw.

CAUTION `pydevd.settrace()` blockiert die Ausführung, wenn Eclipse und der Pydev-Debug-Server nicht gestartet wurden.

Um die letzten beiden Schritte abzudecken: die `o<pydevd>`-Prozedur hilft, aus dem MDI-Modus in den Debugger zu gelangen. Siehe auch die Funktion `call_pydevd` in `util.py` und ihre Verwendung in `remap.involute`, um einen Haltepunkt zu setzen.

Hier ist ein Bildschirmfoto von Eclipse/PyDevd beim Debuggen der obigen `involute`-Prozedur:



Siehe den Python-Code in `configs/sim/axis/remap/getting-started/python` für Details.

9.7.13. Achsenvorschau und geänderte Codeausführung

Für eine vollständige Vorschau des Werkzeugpfads eines umgewandelten Codes müssen einige Vorsichtsmaßnahmen getroffen werden. Um zu verstehen, was vor sich geht, lassen Sie uns den Vorschau- und Ausführungsprozess überprüfen (dies betrifft den AXIS-Fall, aber andere sind ähnlich):

Zunächst ist zu beachten, dass es sich um **zwei** unabhängige Interpreter-Instanzen handelt:

- Ein Beispiel ist das Programm milltask, das ein Programm ausführt, wenn Sie die Schaltfläche "Start" drücken, und die Maschine tatsächlich in Bewegung setzt.
- Eine zweite Instanz in der Benutzeroberfläche, deren Hauptzweck es ist, die Werkzeugwegvorschau zu erzeugen. Diese Instanz "führt" ein Programm aus, sobald es geladen ist, führt aber keine Maschinenbewegungen aus.

Nehmen wir nun an, dass Ihr Neuzuordnungsverfahren einen G38-Tastvorgang enthält, z. B. als Teil eines Werkzeugwechsels mit automatischer Werkzeuglängenabtastung. Wenn der Messtaster fehlschlägt, wäre das eindeutig ein Fehler, und Sie würden eine Meldung anzeigen und das Programm abbrechen.

Wie sieht es nun mit der Vorschau dieses Verfahrens aus? Zum Zeitpunkt der Vorschau ist natürlich nicht bekannt, ob die Sondierung erfolgreich war oder fehlgeschlagen ist - aber Sie würden wahrscheinlich sehen wollen, wie groß die maximale Tiefe der Sondierung ist, und davon ausgehen, dass sie erfolgreich war und die Ausführung fortsetzen, um eine Vorschau weiterer Bewegungen zu erhalten. Außerdem macht es keinen Sinn, eine Meldung "Probe fehlgeschlagen" anzuzeigen und **während der Vorschau** abzubrechen.

Sie können dieses Problem lösen, indem Sie in Ihrer Prozedur testen, ob sie im Vorschau- oder Ausführungsmodus ausgeführt wird. Dies kann durch das Testen von `#<_task>` predefined named parameter überprüft werden - es wird 1 während der tatsächlichen Ausführung und 0 während der Vorschau sein. Siehe *configs/sim/axis/remap/manual-toolchange-with-tool-length-switch/nc_subroutines/manual_change.ngc* für ein vollständiges Anwendungsbeispiel.

In Embedded Python kann die `task`-Instanz durch den Test *self.task* überprüft werden - dies wird in der milltask-Instanz 1 und in der/den preview-Instanz(en) 0 sein.

9.7.14. Neu zuordbare Codes

Vorhandene Codes, die neu zugeordnet werden können

Der derzeitige Satz von **bestehenden** Codes, die neu definiert werden können, ist:

- T*x* (Prepare)
- M6 (Werkzeug wechseln)
- M61 (Werkzeugnummer einstellen)
- M0 (ein laufendes Programm vorübergehend unterbrechen)
- M1 (vorübergehendes Anhalten eines laufenden Programms, wenn der optionale Stoppschalter eingeschaltet ist)
- M7 (aktiviere Kühlnebel)
- M8 (Kühlmittel-Flut aktivieren)
- M9 (Kühlmittelnebel und -flut deaktivieren)
- M60 (Paletten-Shuttles austauschen und dann ein laufendes Programm vorübergehend unterbrechen)
- M62 .. M65 (digitale Ausgangssteuerung)
- M66 (Warten auf Eingabe)
- M67, M68 (Steuerung des Analogausgangs)
- S (Spindeldrehzahl einstellen)
- F (Vorschub einstellen)

Derzeit nicht zugewiesene M-Codes:

Derzeit nicht zugewiesene G-Codes (für remapping) müssen aus den leeren Bereichen der folgenden Tabellen ausgewählt werden. Alle aufgelisteten G-Codes sind bereits in der aktuellen Implementierung von LinuxCNC definiert und kann **nicht** verwendet werden, um neue G-Codes zu remappen. (Entwickler, die neue G-Codes zu LinuxCNC hinzufügen werden ermutigt, auch ihre neuen G-Codes zu diesen Tabellen hinzufügen.)

Table 45. Tabelle der zugewiesenen G-Codes 00-09

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
00	G00									
01	G01									
02	G02									
03	G03									
04	G04									
05	G05	G05.1	G05.2	G05.3						
06										
07	G07									
08	G08									
09										

Table 46. Tabelle der zugewiesenen G-Codes 10-19

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
10	G10									
11										
12										
13										
14										
15										
16										
17	G17	G17.1								
18	G18	G18.1								
19	G19	G19.1								

Table 47. Tabelle der zugewiesenen G-Codes 20-29

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
20	G20									
21	G21									
22										
23										
24										
25										
26										
27										
28	G28	G28.1	G28.2							
29										

Table 48. Tabelle der zugewiesenen G-Codes 50-59

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
30	G30	G30.1								
31										
32										
33	G30	G30.1								
34										
35										
36										
37										
38.										
39.										

Table 49. Tabelle der zugewiesenen G-Codes 50-59

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
40	G40									
41	G41	G41.1								
42	G42	G42.1								
43	G43	G43.1								
44.										
45.										
46										
47										
48										
49	G40									

Table 50. Tabelle der zugewiesenen G-Codes 50-59

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
50										
51										
52										
53	G53									
54	G54									
55	G55									
56	G56									
57	G57									
58	G58									
59	G59	G59.1	G59.2	G59.3						

Table 51. Tabelle der zugewiesenen G-Codes 60-69

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
60	G60									
61	G61	G61.1								
62										
63										
64	G64									
65										
66										
67										
68										
69										

Table 52. Tabelle der zugewiesenen G-Codes 70-79

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
70	G70									
71	G71	G71.1	G71.2							
72	G72	G72.1	G72.2							
73										
74										
75										
76	G76									
77										
78										
79										

Table 53. Tabelle der zugewiesenen G-Codes 80-89

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
80	G80									
81	G81									
82	G82									
83	G83									
84	G84									
85	G85									
86	G86									
87	G87									
88	G88									
89	G89									

Table 54. Tabelle der zugewiesenen G-Codes 90-99

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
90	G90	G90.1								
91	G91	G91.1								
92	G92	G92.1	G92.2	G92.3						
93	G93									
94	G94									
95	G95									
96	G96									
97	G97									
98	G98									
99	G99									

Derzeit nicht zugewiesene M-Codes:

Diese M-Codes sind derzeit undefiniert in der aktuellen Implementierung von LinuxCNC und können verwendet werden, um neue M-Codes zu definieren. (Entwickler, die neue M-Codes in LinuxCNC definieren, werden aufgefordert, sie aus dieser Tabelle zu entfernen.)

Table 55. Tabelle der nicht zugeordneten M-Codes 00-99

#	Mx0	Mx1	Mx2	Mx3	Mx4	Mx5	Mx6	Mx7	Mx8	Mx9
00-09										
10-19	M10	M11	M12	M13	M14	M15	M16	M17	M18	
20-29	M20	M21	M22	M23	M24	M25	M26	M27	M28	M29
30-39		M31	M32	M33	M34	M35	M36	M37	M38	M39
40-49	M40	M41	M42	M43	M44	M45	M46	M47		
50-59					M54	M55	M56	M57	M58	M59
60-69										
70-79					M74	M75	M76	M77	M78	M79
80-89	M80	M81	M82	M83	M84	M85	M86	M87	M88	M89
90-99	M90	M91	M92	M93	M94	M95	M96	M97	M98	M99

Alle M-Codes von `M100` bis `M199` sind bereits benutzerdefinierte M-Codes, die nicht neu zugeordnet werden sollten.

Alle M-Codes von `M200` bis `M999` sind für die Neuzuordnung verfügbar.

9.7.15. Eine kurze Übersicht über die LinuxCNC-Programmausführung

Um die Neuzuordnung von Codes zu verstehen, könnte es hilfreich sein, sich einen Überblick über die Ausführung von `task` und Interpreter zu verschaffen, soweit sie mit der Neuzuordnung zusammenhängt.

Zustand des Interpreters

Konzeptionell besteht der Zustand des Interpreters aus Variablen, die in die folgenden Kategorien fallen:

1. *Konfigurations-Informationen* (typischerweise aus der INI-Datei)
2. *Das "Weltmodell"* - eine Darstellung des aktuellen Maschinenzustands
3. *Modaler Zustand und Einstellungen* - bezieht sich auf den Zustand, der zwischen der Ausführung einzelner NGC-Codes "übertragen" wird - zum Beispiel bleibt die Spindel nach dem Einschalten und der Einstellung der Geschwindigkeit auf dieser Einstellung, bis sie ausgeschaltet wird. Dasselbe gilt für viele Codes wie Vorschub, Einheiten, Bewegungsmodi (Vorschub oder Eilgang) und so weiter.
4. *Interpreter-Ausführungsstatus* - Enthält Informationen über den aktuell ausgeführten Block, ob wir uns in einer Subroutine befinden, Interpreter-Variablen usw. . Der größte Teil dieses Zustands ist in einer - ziemlich unsystematischen - `Struktur _setup` zusammengefasst (siehe interp_internals.hh).

Interaktion zwischen Task und Interpreter, Warteschlange (engl. queue) und Vorauslesen (engl. read-ahead)

Die `task` Teil von LinuxCNC ist verantwortlich für die Koordinierung der tatsächlichen Maschine Befehle - Bewegung, HAL Interaktionen und so weiter. Er verarbeitet nicht selbst die RS274NGC-Sprache. Um dies zu tun, ruft `task` den Interpreter auf, um den nächsten Befehl zu analysieren und auszuführen - entweder von MDI oder der aktuellen Datei.

Die Interpreter-Ausführung erzeugt kanonische Maschinenoperationen, die tatsächlich etwas bewegen. Diese werden jedoch nicht sofort ausgeführt, sondern in eine Warteschlange gestellt. Die eigentliche Ausführung dieser Codes geschieht in der `task` Teil von LinuxCNC: Kanon Befehle sind aus dieser Interpreter Warteschlange gezogen, und ausgeführt, was zu tatsächlichen Bewegungen der Maschine führt.

Das bedeutet, dass der Interpreter in der Regel der tatsächlichen Ausführung von Befehlen weit voraus ist - das Parsen des Programms kann durchaus abgeschlossen sein, bevor eine spürbare Bewegung einsetzt. Dieses Verhalten wird als "Read-ahead" bezeichnet.

Vorhersagen der Maschinenposition

Um die kanonischen Maschinenoperationen beim Vorlesen im Voraus zu berechnen, muss der Interpreter in der Lage sein, die Maschinenposition nach jeder Zeile G-Code vorherzusagen, und das ist nicht immer möglich.

Schauen wir uns ein einfaches Beispielprogramm an, das relative Züge ausführt (G91), und nehmen wir an, die Maschine startet bei x=0,y=0,z=0. Relative Züge bedeuten, dass das Ergebnis des nächsten Zuges von der Position des vorherigen abhängt:

```
N10 G91
N20 G0 X10 Y-5 Z20
N30 G1 Y20 Z-5
N40 G0 Z30
N50 M2
```

Hier kann der Interpreter die Maschinenpositionen für jede Linie eindeutig vorhersagen:

Nach N20: x=10 y=-5 z=20; nach N30: x=10 y=15 z=15; nach N40: x=10 y=15 z=45 und kann so das gesamte Programm parsen und kanonische Operationen im Voraus erzeugen.

Queue-Busters verhindern die Positionsvorhersage

Ein vollständiges Vorlesen ist jedoch nur möglich, wenn der Interpreter die Auswirkungen auf die Position für **jede** Zeile im Programm im Voraus vorhersagen kann. Schauen wir uns ein modifiziertes Beispiel an:

```
N10 G91
N20 G0 X10 Y-5 Z20
N30 G38.3 Z-10
N40 O100 if [#5070 EQ 0]
N50    G1 Y20 Z-5
```

```
N60 O100 else
N70    G0 Z30
N80 O100 endif
N90 G1 Z10
N95 M2
```

Um die Bewegung in N90 vorauszuberechnen, müsste der Interpreter wissen, wo sich die Maschine nach Zeile N80 befindet - und das hängt davon ab, ob der Probe-Befehl erfolgreich war oder nicht, was erst nach der tatsächlichen Ausführung bekannt ist.

Daher sind einige Operationen mit einer weiteren Vorauslesefunktion unvereinbar. Diese werden *queue busters* genannt und sind:

- Lesen des Wertes eines HAL-Pins mit M66: Wert des HAL-Pins nicht vorhersehbar.
- Laden eines neuen Werkzeugs mit M6: Werkzeuggeometrie nicht vorhersehbar.
- Ausführen einer Probe mit G38.*n*: Endposition und Erfolg/Misserfolg nicht vorhersehbar.

Wie Queue-Buster behandelt werden

Wann immer der Interpreter auf einen Queue-Buster stößt, muss er das Vorlesen stoppen und warten, bis das entsprechende Ergebnis vorliegt. Dies funktioniert folgendermaßen:

- Wenn ein solcher Code angetroffen wird, gibt der Interpreter einen speziellen Rückgabewert an `task` zurück (*INTERP_EXECUTE_FINISH*).
- Dieser Rückgabewert signalisiert `task`, das Vorlesen vorerst zu stoppen, alle bisher in der Warteschlange stehenden kanonischen Befehle auszuführen (einschließlich des letzten, d.h. dem "Queue Buster") und dann "den Interpreterzustand mit dem Weltmodell zu synchronisieren". Technisch gesehen bedeutet dies, dass interne Variablen aktualisiert werden, um HAL-Pin-Werte zu reflektieren, Werkzeuggeometrien nach einem M6 neu zu laden und die Ergebnisse einer Probe zu übermitteln.
- Die *synch()*-Methode des Interpreters wird von `task` aufgerufen und tut genau das - alle *aktuellen* Werte des Weltmodells lesen, die für die weitere Ausführung relevant sind.
- An diesem Punkt macht `task` weiter und ruft den Interpreter auf, um weiter zu lesen - bis entweder das Programm endet oder ein anderer Queue-Buster auftaucht.

Wortfolge und Ausführungsreihenfolge

Ein oder mehrere "Wörter" können in einem NGC-"Block" vorhanden sein, wenn sie kompatibel sind (einige schließen sich gegenseitig aus und müssen in verschiedenen Zeilen stehen). Das Ausführungsmodell schreibt jedoch eine strenge Reihenfolge der Ausführung von Codes vor, unabhängig von ihrem Auftreten in der Quellzeile (G-code Ausführungs-Reihenfolge).

Parsen (engl. parsing, für die Interpretation des Quellcodes)

Sobald eine Zeile gelesen wird (entweder im MDI-Modus oder aus der aktuellen NGC-Datei), wird sie geparst, und Flags und Parameter werden in einem "struct block" (struct _setup, member block1) gesetzt.

Diese Struktur enthält alle Informationen über die aktuelle Quellzeile, jedoch unabhängig von der Reihenfolge der Codes in der aktuellen Zeile: Solange mehrere Codes kompatibel sind, führt jede Quellreihenfolge dazu, dass die gleichen Variablen im Strukturblock gesetzt werden. Direkt nach dem Parsen werden alle Codes eines Blocks auf Kompatibilität geprüft.

Ausführung

Nach erfolgreichem Parsen wird der Block mit execute_block() ausgeführt, wobei die verschiedenen Elemente in der Reihenfolge ihrer Ausführung behandelt werden.

Wird ein "Queue Buster" gefunden, wird ein entsprechendes Flag im Interpreterstatus gesetzt (toolchange_flag, input_flag, probe_flag), und der Interpreter gibt einen INTERP_EXECUTE_FINISH-Rückgabewert zurück, der dem Aufrufer (`task`) signalisiert, dass die Vorauslesung vorerst gestoppt und neu synchronisiert wird. Werden nach der Ausführung aller Elemente keine Queue-Buster gefunden, wird INTERP_OK zurückgegeben, was bedeutet, dass die Vorauslesung fortgesetzt werden kann.

Wenn das Lesen nach der Synchronisierung (via der Funktion synch()) fortgesetzt wird, beginnt `task` wieder mit der Ausführung von read()-Operationen des Interpreters. Beim nächsten Lesevorgang werden die oben genannten Flags überprüft und die entsprechenden Variablen gesetzt (da gerade eine Synchronisation ausgeführt wurde, sind die Werte jetzt aktuell). Das bedeutet, dass der nächste Befehl bereits in dem richtig gesetzten Variablenkontext ausgeführt wird.

Prozedurausführung

O-Wort-Prozeduren erschweren die Handhabung von Warteschlangen-Bustern ein wenig. Ein Queue Buster könnte irgendwo in einer verschachtelten Prozedur gefunden werden, was zu einem halbfertigen Prozeduraufruf führt, wenn INTERP_EXECUTE_FINISH zurückgegeben wird. Task stellt sicher, dass das Weltmodell synchronisiert wird und setzt das Parsen und die Ausführung fort, solange noch eine Prozedur ausgeführt wird (call_level > 0).

Wie der Werkzeugwechsel derzeit funktioniert

Die Aktionen, die in LinuxCNC passieren, sind ein bisschen kompliziert, aber es ist notwendig, um die allgemeine Idee zu bekommen, was derzeit passiert, bevor Sie sich auf den Weg machen, um diese Abläufe an Ihre eigenen Bedürfnisse anzupassen.

Beachten Sie, dass durch die Neuzuordnung (engl. remapping) eines vorhandenen Codes die gesamte interne Verarbeitung dieses Codes vollständig deaktiviert wird. Das bedeutet, dass Sie über Ihr gewünschtes Verhalten hinaus (das wahrscheinlich durch ein NGC O-Word oder eine Python-Prozedur beschrieben wird) die internen Aktionen des Interpreters nachbilden müssen, die zusammen eine vollständige Ersetzung des bestehenden Codes ergeben. Der Prolog- und Epilog-Code ist der richtige Ort, um dies zu tun.

Wie die Informationen über das Werkzeug übermittelt werden

Mehrere Prozesse sind an Werkzeuginformationen "interessiert": `task` und sein Interpreter, sowie die Benutzeroberfläche. Außerdem der *halui*-Prozess.

Die Werkzeuginformationen werden in der Struktur "emcStatus" gespeichert, die von allen Beteiligten

gemeinsam genutzt wird. Eines ihrer Felder ist das Array "toolTable", das die Beschreibung enthält, wie sie aus der Werkzeugtabellendatei geladen wurde (Werkzeugnummer, Durchmesser, vorderer und hinterer Winkel und Ausrichtung für Drehmaschinen, Werkzeugkorrektur-Informationen).

Die maßgebliche Quelle und der einzige Prozess, der tatsächlich Werkzeuginformationen in dieser Struktur "festlegt", ist der Prozess "iocontrol". Alle anderen Prozesse konsultieren diese Struktur lediglich. Der Interpreter besitzt eine lokale Kopie der Werkzeugtabelle.

Wer neugierig ist, kann die aktuelle emcStatus-Struktur mit Python statements abrufen. Die Wahrnehmung des Interpreters über das aktuell geladene Werkzeug wird beispielsweise aufgerufen durch:

```
;py,from interpreter import *
;py,print(this.tool_table[0])
```

Sie müssen LinuxCNC von einem Terminal-Fenster aus gestartet haben, um die Ergebnisse zu sehen.

So funktioniert `T`*x* (Werkzeug vorbereiten)

Interpreter-Aktion bei einem `T`x-Befehl

Der Interpreter wertet lediglich den Parameter toolnumber aus, sucht den entsprechenden tooldata-Index, speichert ihn für später in der Variablen `selected_pocket` und stellt einen Kanon-Befehl (SELECT_TOOL) in die Warteschlange. Siehe *Interp::convert_tool_select* in *src/emc/rs274/interp_execute.cc*.

Task-Aktion auf SELECT_TOOL

Wenn `task` dazu kommt, ein SELECT_TOOL zu bearbeiten, sendet es eine EMC_TOOL_PREPARE Nachricht an den `iocontrol` Prozess, der die meisten werkzeugbezogenen Aktionen in LinuxCNC bearbeitet.

In der derzeitigen Implementierung wartet `task` tatsächlich darauf, dass `iocontrol` die Positionierung des Wechslers abschließt, was m.E. nicht notwendig ist, da es die Idee zunichte macht, dass die Vorbereitung des Wechslers und die Ausführung des Codes parallel laufen können.

Iocontrol-Aktion auf EMC_TOOL_PREPARE

Wenn `iocontrol` den Befehl "Select Pocket" sieht, führt es das entsprechende HAL-Pin-Wackeln aus - es setzt den "tool-prep-number"-Pin, um anzuzeigen, welches Werkzeug als nächstes an der Reihe ist, hebt den "tool-prepare"-Pin an und wartet darauf, dass der "tool-prepared"-Pin auf High geht.

Wenn der Wechsler mit der Meldung "tool-prepared" antwortet, betrachtet er die Vorbereitungsphase als abgeschlossen und signalisiert an `task`, die Aufgabe fortzusetzen. Auch dieses "Warten" ist meiner Meinung nach nicht unbedingt erforderlich.

Erstellung des Prologs und Epilogs für `Tx`

Siehe die Python-Funktionen `prepare_prolog` und `prepare_epilog` in `nc_files/remap_lib/python-stdglue/stdglue.py `.

Wie M6 (Wechselwerkzeug) funktioniert

Sie müssen dies vollständig verstehen, bevor Sie es anpassen können. Es ist sehr wichtig für das Schreiben eines Prolog- und Epilog-Handlers für einen neu zugeordneten M6. Das Remapping eines bestehenden Codes bedeutet, dass Sie die internen Schritte, die normalerweise durchgeführt werden, deaktivieren und sie so weit wie nötig für Ihre eigenen Zwecke replizieren.

Auch wenn Sie mit C nicht vertraut sind, empfehle ich Ihnen, sich den Code von "Interp::convert_tool_change" in "src/emc/rs274/interp_convert.cc" anzusehen.

Interpreteraktion bei einem M6-Befehl

Wenn der Interpreter ein M6 sieht, wird er:

1. prüft, ob ein T-Befehl bereits ausgeführt wurde (testet, ob *settings→selected_pocket* >= 0 ist) und schlägt andernfalls mit der Meldung *Need tool prepared -Txx- for tool change* fehl.
2. prüfen, ob die Fräserradiuskompensation aktiv ist, und meldet in diesem Fall "Werkzeugwechsel mit eingeschalteter Fräserradiuskompensation nicht möglich".
3. die Spindel anhalten, außer wenn die INI-Option "TOOL_CHANGE_WITH_SPINDLE_ON" gesetzt ist.
4. eine schnelle Z-Aufwärtsbewegung erzeugen, wenn die INI-Option "TOOL_CHANGE_QUILL_UP" gesetzt ist.
5. wenn TOOL_CHANGE_AT_G30 gesetzt wurde:
   a. Indexer A, B und C verschieben wenn durchführbar
   b. eine schnelle Bewegung zur G30-Position erzeugen
6. einen CHANGE_TOOL canon-Befehl mit der ausgewählten Tasche als Parameter ausführen. CHANGE_TOOL wird:
   a. eine schnelle Bewegung zur TOOL_CHANGE_POSITION erzeugen, wenn dies in der INI eingestellt ist
   b. eine EMC_TOOL_LOAD-NML-Nachricht in die Warteschlange für `task` stellen.
7. die Nummerierungsparameter 5400-5413 entsprechend dem neuen Werkzeug einstellen
8. Signal an `task`, den Aufruf des Interpreters für Readahead zu beenden, indem INTERP_EXECUTE_FINISH zurückgegeben wird, da M6 ein Queue Buster ist.

Was `task` tut, wenn es einen CHANGE_TOOL-Befehl sieht

Auch hier nicht viel mehr, als die Kontrolle an `iocontrol` zu übergeben, indem man ihm eine EMC_TOOL_LOAD Nachricht sendet und zu warten, bis `iocontrol` sein Ding gemacht hat.

Iocontrol-Aktion auf EMC_TOOL_LOAD

1. Es bestätigt den "Tool-Change"-Pin
2. Es wartet, bis der "Tool-changed"-Pin aktiv wird
3. wenn dies geschehen ist:
   a. deassert "Werkzeugwechsel"
   b. Setzen der Pins `tool-prep-number` und `tool-prep-pocket` auf Null

c. die Funktion *load_tool()* mit der Tasche als Parameter ausführen.

Im letzten Schritt werden die Tooltable-Einträge in der *emcStatus*-Struktur gesetzt. Die tatsächlich durchgeführte Aktion hängt davon ab, ob die INI-Option RANDOM_TOOLCHANGER gesetzt wurde, aber am Ende des Prozesses spiegelt *toolTable[0]* das aktuell in der Spindel befindliche Werkzeug wider.

wenn dies geschehen ist:

1. `iocontrol` signalisiert `task` fortzufahren.
2. `task` weist den Interpreter an, eine *synch()*-Operation auszuführen, um zu sehen, was sich geändert hat.
3. Der Interpreter *synch()* zieht alle benötigten Informationen aus dem Weltmodell, darunter auch die geänderte Werkzeugtabelle.

Von da an hat der Interpreter die vollständige Kenntnis des Weltmodells und liest weiter.

Erstellung des Prologs und Epilogs für `M6`

Siehe die Python-Funktionen `change_prolog` und `change_epilog` in `nc_files/remap_lib/python-stdglue/stdglue.py `.

So funktioniert `M61` (Werkzeugnummer ändern)

`M61` erfordert einen nicht-negativen `Q`-Parameter (Werkzeugnummer). Bei Null bedeutet dies *Werkzeug entladen*, sonst *aktuelle Werkzeugnummer auf Q setzen*.

Bau des Ersatzes für `M61`

Ein Beispiel für eine Python-Neudefinition für `M61` findet sich in der Funktion `set_tool_number` in `nc_files/remap_lib/python-stdglue/stdglue.py `.

9.7.16. Status

1. Das RELOAD_ON_CHANGE Feature ist ziemlich kaputt. Starten Sie nach dem Ändern einer Python-Datei neu.

9.7.17. Änderungen

- Die Methode zur Rückgabe von Fehlermeldungen und Fehlschlägen war früher "self.set_errormsg(text)", gefolgt von "return INTERP_ERROR". Dies wurde durch die bloße Rückgabe einer Zeichenkette aus einem Python-Handler oder einer O-word-Subroutine ersetzt. Dies setzt die Fehlermeldung und bricht das Programm ab. Zuvor gab es keine saubere Möglichkeit, eine Python O-word-Subroutine abzubrechen.

9.7.18. Fehlersuche

Im Abschnitt *[EMC]* der INI-Datei kann der Parameter *DEBUG* geändert werden, um verschiedene Stufen von Debug-Meldungen zu erhalten, wenn LinuxCNC von einem Terminal aus gestartet wird.

```
Debug-Level, 0 bedeutet keine Meldungen. Siehe src/emc/nml_intf/debugflags.h für andere
DEBUG = 0x00000002 # configuration
DEBUG = 0x7FFFDEFF # no interp,oword
DEBUG = 0x00008000 # py only
DEBUG = 0x0000E000 # py + remap + Oword
DEBUG = 0x0000C002 # py + remap + config
DEBUG = 0x0000C100 # py + remap + Interpreter
DEBUG = 0x0000C140 # py + remap + Interpreter + NML msgs
DEBUG = 0x0000C040 # py + remap + NML
DEBUG = 0x0003E100 # py + remap + Interpreter + oword + signals + namedparams
DEBUG = 0x10000000 # EMC_DEBUG_USER1 - trace statements
DEBUG = 0x20000000 # EMC_DEBUG_USER2 - trap into Python debugger
DEBUG = 0x10008000 # USER1, PYTHON
DEBUG = 0x30008000 # USER1,USER2, PYTHON # USER2 will cause involute to try to connect to
pydev
DEBUG = 0x7FFFFFFF # All debug messages
```

9.8. Moveoff-Komponente

Die HAL-Komponente moveoff ist eine reine HAL-Methode zur Implementierung von Offsets. Siehe die Manpage (*$ man moveoff*) für die WICHTIGEN Einschränkungen und Warnungen.

Die moveoff-Komponente wird zum Versetzen von Gelenkpositionen unter Verwendung benutzerdefinierter HAL-Verbindungen verwendet. Die Implementierung einer Offset-while-program-is-paused-Funktionalität wird mit entsprechenden Verbindungen für die Eingangspins unterstützt. Neun Gelenke werden unterstützt.

Die Werte der Achsen-Offset-Pins (offset-in-M) werden kontinuierlich (unter Beachtung der Grenzwerte für Wert, Geschwindigkeit und Beschleunigung) an die Ausgangs-Pins (offset-current-M, pos-plusoffset-M, fb-minusoffset-M) angelegt, wenn beide Freigabe-Eingangs-Pins (apply-offsets und move-enable) TRUE sind. Die beiden Freigabeeingänge sind intern verknüpft. Ein *Warn-Pin* wird gesetzt und eine Meldung ausgegeben, wenn der apply-offsets-Pin während der Anwendung von Offsets deassertiert wird. Der Warn-Pin bleibt TRUE, bis die Offsets entfernt werden oder der apply-offsets-Pin gesetzt wird.

Normalerweise ist der move-enable Pin mit externen Steuerelementen verbunden und der apply-offsets Pin ist mit halui.program.is-paused verbunden (für Offsets nur während der Pause) oder auf TRUE gesetzt (für kontinuierlich angewandte Offsets).

Angewandte Offsets werden *automatisch* auf Null zurückgesetzt (unter Beachtung der Grenzwerte), wenn einer der Freigabeeingänge deaktiviert wird. Die Nullwerttoleranz wird durch den Wert des Epsilon-Eingangspins festgelegt.

Wegpunkte werden aufgezeichnet, wenn die Komponente "moveoff" aktiviert ist. Wegpunkte werden mit den Pins waypoint-sample-secs und waypoint-threshold verwaltet. Wenn der Pin für die Rückverfolgungs-Aktivierung (engl. backgrack-enable) TRUE ist, folgt der automatische Rückweg den aufgezeichneten Wegpunkten. Wenn der für die Wegpunkte verfügbare Speicher erschöpft ist, werden die Offsets eingefroren und der waypoint-limit-Pin wird aktiviert. Diese Einschränkung gilt unabhängig vom Zustand des Backtrack-Enable-Pins. Ein Freigabe-Pin muss deaktiviert werden, um eine Rückkehr zur ursprünglichen (nicht versetzten) Position zu ermöglichen.

Backtracking durch Wegpunkte führt zu *langsameren* Bewegungsraten, da die Bewegungen Punkt-zu-Punkt unter Berücksichtigung der Geschwindigkeits- und Beschleunigungseinstellungen erfolgen. Die Geschwindigkeits- und Beschleunigungsgrenzwerte können dynamisch verwaltet werden, um Versätze jederzeit zu kontrollieren.

Wenn backtrack-enable FALSE ist, wird die automatische Rücklaufbewegung **NICHT** koordiniert, jede Achse kehrt mit ihrer eigenen Geschwindigkeit auf Null zurück. Wenn in diesem Zustand ein kontrollierter Weg gewünscht wird, sollte jede Achse manuell auf Null zurückgeführt werden, bevor ein Freigabe-Pin deaktiviert wird.

Die Pins waypoint-sample-secs, waypoint-threshold und epsilon werden nur ausgewertet, wenn sich die Komponente im Leerlauf befindet.

Der Offset-Applied-Ausgangs-Pin dient zur Anzeige des aktuellen Zustands auf einer grafischen Benutzeroberfläche, so dass die Wiederaufnahme des Programms verwaltet werden kann. Wenn die Offsets nicht Null sind, wenn der apply-offsets-Pin deassertiert wird (z.B. bei der Wiederaufnahme eines Programms während einer Pause), werden die Offsets auf Null zurückgesetzt (unter Beachtung der Grenzwerte) und eine *Fehlermeldung* wird ausgegeben.

CAUTION	Wenn Offsets aktiviert und angewendet werden und die Maschine aus irgendeinem Grund ausgeschaltet wird, ist jede *externe* HAL-Logik, die Aktivierungspins und Offset-in-M-Eingänge verwaltet, für deren Zustand verantwortlich, wenn die Maschine anschließend wieder eingeschaltet wird.

Diese HAL-only Methode für Offset ist LinuxCNC in der Regel nicht bekannt und nicht in der GUI-Vorschau zeigt. **Keine Schutzmaßnahmen verfügbar** für Offset-Bewegungen, wenn sie die von LinuxCNC verwalteten weichen Grenzen überschreiten. Da weiche Grenzen nicht beachtet werden, kann eine Offset-Bewegung auf harte Grenzen stoßen (oder **CRASH**, wenn es keine Endschalter gibt). Die Verwendung der Eingänge offset-min-M und offset-max-M zur Begrenzung des Verfahrwegs wird empfohlen. Das Auslösen einer harten Grenze wird die Maschine ausschalten - siehe **Caution** oben.

Die Offset-in-M-Werte können mit INI-Datei-Einstellungen festgelegt, über eine grafische Benutzeroberfläche gesteuert oder durch andere HAL-Komponenten und Verbindungen verwaltet werden. Feste Werte können in einfachen Fällen geeignet sein, in denen die Richtung und der Betrag des Offsets genau definiert sind, aber eine Kontrollmethode erforderlich ist, um einen Aktivierungs-Pin zu deaktivieren, und so die Offsets auf Null zurückzusetzen. Die grafischen Benutzeroberflächen können dem Benutzer die Möglichkeit bieten, Offset-Werte für jede Achse einzustellen, zu erhöhen, zu verringern und zu akkumulieren, und sie können Offset-in-M-Werte auf Null setzen, bevor ein Freigabe-Pin deaktiviert wird.

Die Standardwerte für accel, vel, min, max, epsilon, waypoint-sample-secs und waypoint-threshold sind möglicherweise nicht für jede Anwendung geeignet. Diese HAL-Komponente kennt keine Grenzen, die an anderer Stelle von LinuxCNC erzwungen werden. Benutzer sollten die Verwendung in einem Simulator-Anwendung zu testen und alle Gefahren vor der Verwendung auf Hardware verstehen.

Sim-Konfigurationen zur Demonstration einer Komponenten und eine Benutzeroberfläche (moveoff_gui) befinden sich in:

- configs/sim/axis/moveoff (axis-ui)

- configs/sim/touchy/ngcgui (touchy-ui)

9.8.1. Ändern einer bestehenden Konfiguration

Eine vom System bereitgestellte HAL-Datei (LIB:hookup_moveoff.tcl) kann verwendet werden, um eine bestehende Konfiguration für die Verwendung der moveoff-Komponente anzupassen. Zusätzliche Einstellungen in der INI-Datei unterstützen die Verwendung einer einfachen Benutzeroberfläche (moveoff_gui) zur Steuerung von Offsets.

Wenn die System-HAL-Datei (LIB:hookup_moveoff.tcl) ordnungsgemäß in einer Konfigurations-INI-Datei angegeben ist, wird sie:

1. die ursprünglichen Pinverbindungen joint.N.motor-pos-cmd und joint.N.motor-pos-fb trennen
2. Die moveoff-Komponente (unter dem Namen mv) mit einem Profil (engl. personality) laden (loadrt), die alle in der INI-Datei angegebenen Achsen aufnehmen kann
3. Funktionen der Auszugskomponenten in der gewünschten Reihenfolge hinzufügen (addf)
4. Pins joint.N.motor-pos-cmd und joint.N.motor-pos-fb erneut verbinden, um die moveoff-Komponente zu verwenden
5. Betriebsparameter und Grenzwerte der moveoff-Komponenten für jede Achse gemäß den zusätzlichen Einstellungen in der INI-Datei festlegen

Note: Die Anwendung moveoff_gui unterstützt Konfigurationen, die bekannte Kinematikmodule mit KINEMATICS_TYPE=KINEMATICS_IDENTITY verwenden. Zu den unterstützten Modulen gehören: trivkins. Bei Identitätskinematiken weist moveoff_gui jeden Achsennamen, der mit dem Kommandozeilenparameter *-axes axisnames* angegeben wird, dem entsprechenden Gelenk zu.

Ändern Sie eine bestehende Konfiguration wie folgt:

Vergewissern Sie sich, dass es einen INI-Datei-Eintrag für `[HAL]HALUI` gibt und erstellen Sie einen neuen `[HAL]HALFILE`-Eintrag für LIB:hookup_moveoff.tcl. Der Eintrag für LIB:hookup_moveoff.tcl sollte allen `HALFILE=`-Einträgen für HAL-Dateien folgen, die Pins für `joint.__N__.motor-pos-cmd`, `joint.__N__.motor-pos-fb` und alle an diese Pins angeschlossenen Komponenten (z. B. PID- und Encoder-Komponenten in einem Servosystem) verbinden.

```
[HAL]
HALUI   = halui
HALFILE = existing_configuration_halfile_1
# ...
HALFILE = existing_configuration_halfile_n
HALFILE = LIB:hookup_moveoff.tcl
```

Fügen Sie INI-Datei-Einträge für die Einstellungen pro Achse für jede verwendete Achse hinzu (wenn ein Eintrag nicht definiert ist, wird der entsprechende Eintrag aus dem Abschnitt [AXIS_n] verwendet, wird kein Eintrag gefunden, so wird die Standardeinstellung der moveoff-Komponente verwendet).

NOTE	Es wird NICHT empfohlen, die Komponentenvorgaben oder die Werte des Abschnitts [AXIS_n] für die Offset-Einstellungen der einzelnen Achsen zu verwenden.

```
[MOVEOFF_n]
MAX_LIMIT =
MIN_LIMIT =
MAX_VELOCITY =
MAX_ACCELERATION =
```

Fügen Sie INI-Datei-Einträge für die Einstellungen der moveoff-Komponent hinzu (um Standardeinstellungen für moveoff zu vermeiden):

```
[MOVEOFF]
EPSILON =
WAYPOINT_SAMPLE_SECS =
WAYPOINT_THRESHOLD =
```

Das moveoff_gui wird verwendet, um zusätzliche erforderliche Verbindungen herzustellen und eine Popup-GUI zu erstellen:

1. Aktivieren/Deaktivieren von Offsets über eine Umschalttaste (engl. togglebutton).
2. Bereitstellung einer Schaltfläche zum Aktivieren/Deaktivieren des Backtrackings
3. Steuertasten zum Inkrementieren/Dekrementieren/Nullstellen jeder Achsenverschiebung
4. Anzeige des aktuellen Wertes jeder Achsenverschiebung
5. Anzeige des aktuellen Offset-Status (deaktiviert, aktiv, entfernt, etc.)

Die bereitgestellten Schaltflächen sind optional und hängen vom Zustand des moveoff-Komponenten-Pins move-enable ab. Wenn der Pin mv.move-enable beim Start des moveoff_gui NICHT angeschlossen ist, werden sowohl eine Anzeige als auch Steuerelemente zur Aktivierung des Offsets bereitgestellt. In diesem Fall verwaltet der moveoff_gui den moveoff component move-enable pin (mv.move-enable) sowie die Offsets (mv.move-offset-in-M) und die Backtracking-Freigabe (mv.backtrack-enable)

Wenn der mv.move-enable-Pin beim Starten des moveoff_gui angeschlossen ist, bietet die moveoff_gui eine Anzeige, aber KEINE Steuerung. Dieser Modus unterstützt Konfigurationen, die ein Jogwheel oder andere Methoden zur Steuerung der Offset-Eingänge und der Enable-Pins verwenden (mv.offset-in-M, mv.move-enable, mv.backtrack-enable).

Der moveoff_gui stellt die erforderlichen Verbindungen für die Pins der moveoff-Komponente her: mv.power_on und mv.apply-offsets. Der mv.power_on-Pin wird mit dem motion.motion-enabled-Pin verbunden (ein neues Signal wird automatisch erstellt, falls erforderlich). Der mv.apply-offsets ist mit halui.program.is-paused verbunden oder auf 1 gesetzt, je nach der Kommandozeilenoption -mode [onpause | always]. Bei Bedarf wird automatisch ein neues Signal erzeugt.

Um das moveoff_gui zu verwenden, fügen Sie in der INI-Datei [APPLICATIONS] einen Eintrag wie folgt hinzu:

```
[APPLICATIONS]
# Hinweis: eine Verzögerung (in Sekunden) kann erforderlich sein, wenn Verbindungen
# über Post-GUI HAL-Dateien ([HAL]POSTGUI_HALFILE=) hergestellt werden.
DELAY = 0
APP = moveoff_gui option1 option2 ...
```

Wird die HAL-Datei LIB:hookup_moveoff.tcl zum Laden und Anschließen der moveoff-Komponente verwendet, so wird der mv.move-enable-Pin nicht angeschlossen und die vom moveoff_gui bereitgestellten lokalen Steuerungen werden verwendet. Dies ist die einfachste Methode, um die moveoff-Komponente zu testen oder zu demonstrieren, wenn eine bestehende INI-Konfiguration geändert wird.

Um externe Steuerungen zu aktivieren und gleichzeitig die moveoff_gui-Anzeige für Offset-Werte und Status zu verwenden, müssen HAL-Dateien, die auf LIB:hookup_moveoff.tcl folgen, zusätzliche Verbindungen herstellen. Die mitgelieferten Demonstrationskonfigurationen (configs/sim/axis/moveoff/*.ini) verwenden beispielsweise eine einfache System-HAL-Datei (namens LIB:moveoff_external.hal), um die Pins mv.move-enable,mv.offset-in-M und mv.bactrack-enable mit Signalen zu verbinden:

```
[HAL]
HALUI = halui
# ...
HALFILE = LIB:hookup_moveoff.tcl
HALFILE = LIB:moveoff_external.hal
```

Die von LIB:moveoff_external.hal hergestellten Verbindungen (für eine dreiachsige Konfiguration) sind:

```
net external_enable mv.move-enable

net external_offset_0 mv.offset-in-0
net external_offset_1 mv.offset-in-1
net external_offset_2 mv.offset-in-2

net external_backtrack_en mv.backtrack-enable
```

Diese Signale (external_enable, external_offset_M, external_backtrack_en) können von nachfolgenden HALFILES (einschließlich POSTGUI_HALFILEs) verwaltet werden, um eine angepasste Steuerung der Komponente zu ermöglichen, während die moveoff_gui-Anzeige für aktuelle Offset-Werte und den Offset-Status verwendet wird.

Der moveoff_gui wird mit Kommandozeilenoptionen konfiguriert. Einzelheiten zur Funktionsweise von moveoff_gui finden Sie in der Manpage:

```
$ man moveoff_gui
```

Eine kurze Auflistung der Kommandozeilenoptionen für moveoff_gui finden Sie in der Kommandozeilenoption help:

```
$ moveoff_gui --help

Usage:
moveoff_gui [Optionen]

Optionen:
    [--help | -? | -- -h ] (Dieser Hilfe-Text)
```

```
    [-mode [onpause | always]] (Voreinstellung: onpause)
                                   (onpause: zeigt die Benutzeroberfläche, wenn das
Programm pausiert)
                                   (always: Benutzeroberfläche immer anzeigen)

    [-axes axisnames] (Standard: xyz (ohne Leerzeichen))
                               (Buchstaben aus der Menge von: x y z a b c u v w)
                               (Beispiel: -axes z)
                               (Beispiel: -axes xz)
                               (Beispiel: -axes xyz)
    [-inc Inkrementwert] (Voreinstellung: 0.001 0.01 0.10 1.0 )
                               (geben Sie einen pro -inc an (bis zu 4) )
                               (Beispiel: -inc 0.001 -inc 0.01 -inc 0.1 )
    [-size ganze Zahl] (Voreinstellung: 14)
                               (Die Gesamtgröße des Popup-Fensters der Benutzeroberfläche
basiert auf der Schriftgröße)
    [-loc center|+x+y] (Voreinstellung: center)
                               (Beispiel: -loc +10+200)
    [-autoresume] (Voreinstellung: nicht verwendet)
                               (Programm fortsetzen, wenn move-enable deaktiviert wird)
    [-delay delay_secs] (Voreinstellung: 5 (Wiederaufnahmeverzögerung))

Optionen für Sonderfälle:
    [-noentry] (Voreinstellung: nicht verwendet)
                              (keine Eintrags-Widgets erstellen)
    [-no_resume_inhibit] (Voreinstellung: nicht verwendet)
                              (keinen resume-inhibit-pin verwenden)
    [-no_pause_requirement] (Voreinstellung: nicht verwendet)
                              (keine Prüfung auf halui.program.is-paused)
    [-no_cancel_autoresume] (Voreinstellung: nicht verwendet)
                              (nützlich für die Rücknahme von Offsets mit einfachen)
                              (externe Steuerung)
    [-no_display] (Voreinstellung: nicht verwendet)
                              (Verwendung, wenn sowohl externe Steuerungen als auch
Anzeigen)
                              (verwendet werden (siehe Hinweis))

Hinweis: Wenn der moveoff move-enable Pin (mv.move-enable) angeschlossen ist während
      moveoff_gui gestartet wird, sind externe Steuerungen erforderlich und nur
      die Bildschirm-Anzeigen sind verfügbar.
```

9.9. Eigenständiger Interpreter

Der eigenständige Interpreter rs274 kann über die Kommandozeile verwendet werden.

9.9.1. Anwendung

```
Usage: rs274 [-p interp.so] [-t tool.tbl] [-v var-file.var] [-n 0|1|2]
          [-b] [-s] [-g] [input file [output file]]

    -p: Specify the pluggable interpreter to use
    -t: Specify the .tbl (tool table) file to use
    -v: Specify the .var (parameter) file to use
    -n: Specify the continue mode:
```

```
           0: continue
           1: enter MDI mode
           2: stop (default)
    -b: Toggle the 'block delete' flag (default: OFF)
    -s: Toggle the 'print stack' flag (default: OFF)
    -g: Toggle the 'go (batch mode)' flag (default: OFF)
    -i: specify the .ini file (default: no ini file)
    -T: call task_init()
    -l: specify the log_level (default: -1)
```

9.9.2. Beispiel

Um die Ausgabe einer Schleife zu sehen, können wir zum Beispiel rs274 in der folgenden Datei ausführen und sehen, dass die Schleife nie endet. Um die Schleife zu verlassen, drücken Sie Strg-Z. Die folgenden zwei Dateien werden benötigt, um dieses Beispiel auszuführen.

test.ngc

```
#<test> = 123.352

o101 while [[#<test> MOD 60 ] NE 0]
(debug,#<test>)
    #<test> = [#<test> + 1]
o101 endwhile

M2
```

test.tbl

```
T1 P1 Z0.511 D0.125 ;1/8 end mill
T2 P2 Z0.1 D0.0625 ;1/16 end mill
T3 P3 Z1.273 D0.201 ;#7 tap drill
```

Befehl

```
rs274 -g test.ngc -t test.tbl
```

9.10. Offsets der externen Achse

Externe Achsen-Offsets werden bei Teleop (World) Jogs und koordinierten (G-Code) Bewegungen unterstützt. Externe Achsen-Offsets werden pro Achse durch INI-Datei-Einstellungen aktiviert und dynamisch durch HAL-Eingangsstifte gesteuert. Die HAL-Schnittstelle ähnelt der für das Joggen von Rädern verwendeten Schnittstelle. Diese Art von Schnittstelle wird typischerweise mit einem manuellen Impulsgenerator (MPG) implementiert, der mit einer Encoder-HAL-Komponente verbunden ist, die Impulse zählt.

9.10.1. INI-Datei Einstellungen

Für jeden Buchstaben der Achse (**L** in xyzabcuvw):

```
[AXIS_L]OFFSET_AV_RATIO = Wert (steuert Beschleunigung/Drehung für externe Offsets)
```

1. Erlaubte Werte: 0 ⇐ Wert ⇐ 0.9
2. Nicht zulässige Werte werden durch 0.1 mit der Meldung an stdout ersetzt
3. Standardwert: 0 (deaktiviert den externen Offset).
   Folge: Weglassen von [AXIS_L]OFFSET_AV_RATIO deaktiviert den externen Offset für die Achse.
4. Wenn der Wert nicht Null ist, passt das OFFSET_AV_RATIO (**r**) die konventionelle (Planungs-) Höchstgeschwindigkeit und -beschleunigung an, um die [AXIS_L]-Einschränkungen einzuhalten:

```
maximale Planungsgeschwindigkeit = (1-r) * MAX_VELOCITY
externe Offset-Geschwindigkeit = ( r) * MAX_VELOCITY

planning max acceleration    = (1-r) * MAX_ACCELERATION
external offset acceleration = (  r) * MAX_ACCELERATION
```

9.10.2. HAL-Pins

Pro-Achse Bewegungs (engl. Motion)-HAL Pins

Für jeden Achsenbuchstaben (***L*** in xyzabcuvw)

1. **axis.*L*.eoffset-enable** Input(bool): enable
2. **axis.*L*.eoffset-scale** Input(real): scale factor
3. **axis.*L*.eoffset-counts** Input(sint): input to counts register
4. **axis.*L*.eoffset-clear** Input(bool): clear requested offset
5. **axis.*L*.eoffset** Output(real): current external offset
6. **axis.*L*.eoffset-request** Output(real): requested external offset

Andere Bewegungs(engl. Motion)-HAL Pins

1. **motion.eoffset-active** Output(bool): non-zero external offsets applied
2. **motion.eoffset-limited** Output(bool): motion inhibited due to soft limit

9.10.3. Anwendung

Die Achseneingangs-HAL-Pins (enable, scale, counts) ähneln den Pins, die für das Wheel-Jogging verwendet werden.

Offset-Berechnung

In jeder Servoperiode wird der Pin *axis.L.eoffset-counts* mit seinem Wert in der vorherigen Periode verglichen. Die Zunahme oder Abnahme (positives oder negatives Delta) des Pins *axis.L.eoffset-counts* wird mit dem aktuellen Wert des Pins *axis.L.eoffset-scale* multipliziert. Dieses Produkt wird in einem internen Register akkumuliert und an den HAL-Pin *axis.L.eoffset-request* exportiert. Das Akkumulationsregister wird bei jedem Einschalten der Maschine auf Null zurückgesetzt.

Der angeforderte Offset-Wert wird verwendet, um die Bewegung für den Offset zu planen, der auf die *L* -Koordinate angewendet und durch den HAL-Pin *axis.L.eoffset* dargestellt wird. Die geplante Bewegung berücksichtigt die zugewiesenen Geschwindigkeits- und Beschleunigungsbeschränkungen und kann begrenzt werden, wenn die Nettobewegung (Offset plus Teleop-Jogging oder koordinierte Bewegung) eine weiche Grenze für die *L*-Koordinate erreicht.

Bei vielen Anwendungen ist der *axis.L.eoffset-scale*-Pin konstant und die Netto-*axis.L.eoffset-request* -Antwort auf *axis.L.eoffset-counts* entspricht dem Produkt aus dem kumulierten Wert von *axis.L.eoffset-counts* und den (konstanten) *axis.L.eoffset-scale*-Pin-Werten.

Maschine aus/Maschine ein

Wird die Maschine ausgeschaltet, so wird die **aktuelle Position** mit **externen Offsets beibehalten**, damit es keine unerwarteten Bewegungen beim Aus- oder Einschalten gibt.

Bei jedem Start (Einschalten der Maschine) wird das interne Zählregister für jeden HAL-Pin *axis.L.eoffset-counts* auf Null gesetzt und der entsprechende HAL-Ausgangspin *axis.L.eoffset* wird auf Null zurückgesetzt.

Mit anderen Worten: Externe Offsets werden **bei jedem Start** (Maschine ein) als NULL definiert, unabhängig vom Wert der *axis.L.eoffset-counts*-Pins. Um Verwirrung zu vermeiden, wird empfohlen, dass alle *axis.L.eoffset-counts*-Pins auf Null gesetzt werden, wenn die Maschine ausgeschaltet ist.

Weiche Grenzwerte

Externe Achsen-Offset-Bewegungen werden unabhängig mit den durch *[AXIS_L]OFFSET_AV_RATIO* festgelegten Geschwindigkeits- und Beschleunigungseinstellungen geplant. Die Offset-Bewegung wird weder mit dem Teleop-Jogging noch mit der koordinierten (G-Code-) Bewegung koordiniert. Während des Teleop-Jogging und der koordinierten (G-Code-)Bewegung schränken weiche Achsengrenzen (*[AXIS_L]MIN_LIMIT,MAX_LIMIT*) die Bewegung der Achse ein.

Wenn externe Offsets angewendet werden und die Bewegung eine weiche Grenze erreicht (durch externe Offset-Erhöhungen oder Teleop-Jogging oder koordinierte Bewegung), wird der HAL-Pin *motion.eoffset-limited* aktiviert und der Achsenwert nominal auf der weichen Grenze gehalten. Dieser HAL-Pin kann von der zugehörigen HAL-Logik verwendet werden, um weitere E-Offset-Zählungen abzuschneiden oder die Maschine anzuhalten (z. B. durch Anschluss an *halui.machine.off*). Wird die Achse innerhalb des Softlimits bewegt, so wird der *motion.eoffset-limited*-Pin zurückgesetzt.

Beim Betrieb an einer weichen Grenze während einer koordinierten Bewegung, die den geplanten Achsenwert weiter verändert, zeigt der HAL-Ausgangspin *axis.L.eoffset* den aktuellen Offset an - die Distanz, die benötigt wird, um die Grenze zu erreichen, anstatt der berechneten Offset-Anforderung.

Dieser angezeigte Wert ändert sich, wenn sich der geplante Achsenwert ändert.

Der HAL-Pin *axis.L.eoffset-request* zeigt den aktuellen angeforderten Offset an, der das Produkt aus dem internen Zählregister und der eoffset-Skala ist. Im Allgemeinen hinkt der Wert des Pins *axis.L.eoffset* dem Wert von *axis.L.eoffset-request* hinterher, da der externe Offset einer Beschleunigungsgrenze unterliegt. Beim Betrieb an einer weichen Grenze wirken sich zusätzliche Aktualisierungen der_ *'axis.L.eoffset-counts' weiterhin auf den angeforderten externen Offset aus, wie er im axis._L.eoffset-request* -HAL-Pin reflektiert wird.

Beim Teleop-Jogging mit aktivierten externen Offsets **und** angewandten Werten ungleich Null wird bei Erreichen eines Soft-Limits die Bewegung in der betreffenden Achse **ohne Verzögerungsintervall** angehalten. In ähnlicher Weise wird bei einer koordinierten Bewegung mit aktivierten externen Offsets das Erreichen eines Soft-Limits zum Anhalten der Bewegung ohne Verzögerungsphase führen. In diesem Fall spielt es keine Rolle, ob die Offsets Null sind.

Wenn die Bewegung ohne Verzögerungsphase gestoppt wird, können die **Beschleunigungsgrenzen des Systems verletzt werden**, was zu Folgefehlern führt: 1) einem Schleppfehler (und/oder einem Klopfen) bei einem Servomotor-System, 2) einem Verlust von Schritten bei einem Schrittmotor-System. Im Allgemeinen wird empfohlen, externe Offsets so zu verwenden, dass eine Annäherung an die weichen Grenzen vermieden wird.

Anmerkungen

Externe Versätze gelten für Achsenkoordinatenbuchstaben (xyzabcuvw). Alle Gelenke müssen referenziert werden, bevor externe Achsenversätze berücksichtigt werden.

Wird ein *axis.L.eoffset-enable*-HAL-Pin zurückgesetzt, wenn sein Offset ungleich Null ist, wird der Offset beibehalten. Der Offset kann gelöscht werden durch:

1. ein Umschalter "Maschine aus/Maschine an"
2. Reaktivierung des Freigabe-Pins und Inkrementierung/Dekrementierung des HAL-Pins *axis0. L.eoffset-counts*, um den Offset auf Null zu setzen.
3. Pulsieren des HAL-Pins *axis.L.eoffset-clear*

Externe Offsets sind für die Verwendung mit "kleinen" Offsets vorgesehen, die innerhalb der Soft-Limit-Grenzen angewendet werden.

Softlimits werden sowohl beim Teleop-Jogging als auch bei der koordinierten Bewegung beachtet, wenn externe Offsets angewendet werden. Wenn jedoch während einer koordinierten Bewegung eine weiche Grenze erreicht wird, kann sich die Verringerung des externen Offsets **nicht von der weichen Grenze entfernen**, **wenn die geplante Bewegung in dieselbe Richtung fortgesetzt wird**. Dieser Umstand kann auftreten, da die Geschwindigkeit der Korrektur des Offsets (wie mit *[AXIS_L]OFFSET_AV_RATIO* eingestellt) geringer sein kann als die geplante Gegenbewegung. In solchen Fällen wird die geplante koordinierte Bewegung **angehalten** (oder gestoppt), um eine Bewegung weg von der weichen Grenze zu ermöglichen, wenn korrigierende Änderungen am externen Offset vorgenommen werden.

Warnung

Die Verwendung von externen Offsets kann die Maschinenbewegung erheblich verändern. Die Steuerung von externen Offsets mit HAL-Komponenten und Verbindungen sowie alle zugehörigen Benutzerschnittstellen sollten vor dem Einsatz sorgfältig entworfen und getestet werden.

9.10.4. Verwandte HAL-Komponenten

eoffset_per_angle.comp

Komponente zur Berechnung eines externen Offsets aus einer Funktion auf der Grundlage eines gemessenen Winkels (Drehkoordinate oder Spindel). Siehe die Manpage für Details (**$ man eoffset_per_angle**).

9.10.5. Testen

Der externe Achsenversatz wird durch Hinzufügen einer *[AXIS_L]*-Einstellung für jede Kandidatenachse aktiviert. Zum Beispiel:

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.2
```

Zum Testen ist es praktisch, eine Jogwheel-Schnittstelle mit der **sim_pin**-GUI zu simulieren. Zum Beispiel in einem Terminal:

```
$ sim_pin axis.z.eoffset-enable axis.z.eoffset-scale axis.z.eoffset-counts
```

Die Verwendung externer Offsets wird durch die Anzeige von Informationen zu den aktuellen Offsets unterstützt: der aktuelle E-Offset-Wert und der angeforderte E-Offset-Wert, der Achsen-Pos-Cmd und (für eine Identitätskinematik-Maschine) der entsprechende Gelenkmotor-Pos-Cmd und Motor-Offset. Die mitgelieferte Sim-Konfiguration (siehe unten) demonstriert ein Beispiel für ein PyVCP-Panel für die AXIS GUI.

Wenn keine benutzerdefinierte Anzeige vorhanden ist, kann **halshow** als Hilfsanwendung mit einer benutzerdefinierten Überwachungsliste gestartet werden.

Beispiel für INI-Datei-Einstellungen zur Simulation der HAL-Pin-Eoffset-Verbindungen und zur Anzeige von E-Offset-Informationen für die Z-Achse (für Identitätskinematik mit z==joint2):

```
[APPLICATIONS]
APP = sim_pin \
      axis.z.eoffset-enable \
      axis.z.eoffset-scale \
      axis.z.eoffset-counts \
      axis.z.eoffset-clear

APP = halshow --fformat "%0.5f" ./z.halshow
```

Wo sich die Datei z.halshow (im Konfigurationsverzeichnis) befindet:

```
pin+joint.2.motor-pos-cmd
pin+joint.2.motor-offset
pin+axis.z.pos-cmd
pin+axis.z.eoffset
pin+axis.z.eoffset-request
pin+motion.eoffset-limited
```

9.10.6. Beispiele

Die bereitgestellten Simulationskonfigurationen demonstrieren die Verwendung externer Offsets, um einen Ausgangspunkt für die Anpassung an die reale Hardware zu bieten.

Die Sim-Konfigurationen verwenden die INI-Einstellung *[HAL]HALFILE =LIB:basic_sim.tcl*, um alle Routine-HAL-Verbindungen für die in der INI-Datei *[TRAJ]COORDINATES*= angegebenen Achsen. Die HAL-Logik, die zur Demonstration der externen Offset-Funktionalität benötigt wird und die GUI HAL Pin Verbindungen für ein PyVCP Panel sind in getrennten HAL Dateien. Eine Nicht-Simulations-Konfiguration sollte den Eintrag *LIB:basic_sim.tcl* für die Maschine in den HALFILEs ersetzen. Die mitgelieferten PyVCP-Dateien (.hal und .xml) können ein Ausgangspunkt für anwendungsspezifische Benutzeroberflächen sein.

eoffsets.ini

Die Sim-Konfiguration *sim/configs/axis/external_offsets/eoffsets.ini* demonstriert eine kartesische XYZ-Maschine mit Steuerelementen zur Aktivierung externer Offsets auf jeder Achse.

Alle wichtigen Positions- und Offsetwerte werden angezeigt.

Ein sim_pin GUI bietet Steuerelemente für die Achsen-Offset-Pins: eoffset-scale & eoffset-counts (über Signal e:<L>counts), eoffset-clear (über Signal e:clearall)

Ein Skript (eoffsets_monitor.tcl) wird verwendet, um die *axis.L.counts*-Pins beim Ausschalten der Maschine auf Null zu setzen.

jwp_z.ini

Die Sim-Konfiguration *sim/configs/axis/external_offsets/jwp_z.ini* demonstriert eine Jog-While-Pause-Funktion für eine einzelne (Z-)Koordinate:

Die LEDs auf dem Bedienfeld dienen zur Anzeige wichtiger Statusinformationen.

Es gibt Steuerelemente zum Einstellen des Skalierungsfaktors für den Eoffset und zum Erhöhen/Verringern/Löschen der Eoffset-Zähler.

dynamische_offsets.ini

Diese Sim-Konfiguration *sim/configs/axis/external_offsets/dynamic_offsets.ini* demonstriert dynamisch angewandte Offsets durch Anschluss einer Sinuswellenform an die externen Offset-Eingänge für die Z-

Koordinate.

Die LEDs auf dem Bedienfeld dienen zur Anzeige wichtiger Statusinformationen.

Es gibt Steuerelemente zur Änderung der INI-Datei-Einstellungen für die maximale Geschwindigkeit und Beschleunigung der Z-Achse.

Die Parameter des Wellenformgenerators lassen sich mit Hilfe von Reglern einstellen.

Eine Halscope-App wird gestartet, um die angelegte Wellenform, die Offset-Antwort und die Motor-CMD-Antwort anzuzeigen.

NOTE	Änderungen an der z-Koordinate max-acceleration und max-velocity werden während der Ausführung eines Programms nicht bestätigt.

opa.ini (eoffset_per_angle)

Die opa.ini-Konfiguration verwendet die INI-Komponente eoffset_per_angle (**$ man eoffset_per_angle**), um eine XZC-Maschine mit funktionalen Offsets zu demonstrieren, die aus der C-Koordinate (Winkel) berechnet und auf die Transversalkoordinate (X) angewendet werden. Die Offset-Berechnungen basieren auf einem bestimmten Referenzradius, der in der Regel durch einen M68-Befehl eines Programms (oder MDI) zur Steuerung eines **motion.analog-out-NN**-Pins festgelegt wird.

Die LEDs auf dem Bedienfeld dienen zur Anzeige wichtiger Statusinformationen.

Es werden Funktionen für Innen- und Außenpolygone (nsides >= 3), Sinuswellen und Rechteckwellen bereitgestellt. Die Funktionen können mit dem Stift fmul in der Frequenz multipliziert und mit dem Stift rfrac in der Amplitude verändert werden (Bruchteil des Referenzradius).

Es gibt Bedienelemente zum Starten/Stoppen von Offset-Wellenformen und zum Einstellen des Funktionstyps und seiner Parameter.

9.11. Werkzeugdatenbank-Schnittstelle

Die Werkzeugdaten werden üblicherweise durch eine Werkzeugtabellendatei beschrieben, die durch eine INI-Datei-Einstellung angegeben wird: [EMCIO]TOOL_TABLE=tooltable_filename. Eine Werkzeugtabellendatei besteht aus einer Textzeile für jedes verfügbare Werkzeug, welche die Parameter des Werkzeugs beschreibt, siehe Werkezugtabellen-Format.

Die Werkzeugdatenbankschnittstelle bietet eine alternative Methode zur Beschaffung von Werkzeugdaten über ein separates Programm, das eine Datenbank mit Werkzeugen verwaltet.

9.11.1. Schnittstelle

INI-Datei Einstellungen

Die Einstellungen in der INI-Datei ermöglichen den (optionalen) Betrieb eines vom Benutzer bereitgestellten Werkzeugdatenbankprogramms:

```
[EMCIO]
DB_PROGRAM = db_program [args]
```

Wenn es enthalten ist, gibt **db_program** den Pfad zu einem vom Benutzer bereitgestellten ausführbaren Programm an, das tooldata bereitstellt. Es können bis zu 10 durch Leerzeichen getrennte Args angegeben werden, die beim Start an **db_program** übergeben werden.

NOTE INI-Datei-Einstellungen für [EMCIO]TOOL_TABLE werden ignoriert, wenn ein **db_program** angegeben ist.

NOTE Das **db_program** kann in jeder Sprache implementiert werden, die derzeit von LinuxCNC unterstützt wird (z. B. Bash-Skripte, Python oder Tcl-Skripte, C/C++-Programme), solange es mit den Schnittstellenmeldungen übereinstimmt, die auf stdin empfangen und auf stdout zurückgegeben werden. Ein **db_program** kann Daten aus einer flachen Datei, einer relationalen Datenbank (z.B. SQLite) oder anderen Datenquellen verwalten.

db_program-Operation (v2.1)

Wenn ein **db_progam** angegeben ist, wird wie folgt vorgegangen:

1. Beim Starten startet LinuxCNC das **db_program** und verbindet sich mit dessen stdin und stdout.
2. Das **db_program** muss mit einer einzeiligen Bestätigung antworten, die aus einem Versionsstring besteht (z.B. "v2.1"). Wenn die Version nicht mit der Version der LinuxCNC-Datenbankschnittstelle kompatibel ist, sind keine Werkzeuge verfügbar.
3. Nach einer erfolgreichen Bestätigung, gibt LinuxCNC einen *g* (**get**) Befehl aus, um alle Werkzeuge anzufordern. Das **db_program** muss mit einer Folge von Antworten antworten, um jedes verfügbare Werkzeug zu identifizieren. Das Textformat der Antworten ist identisch mit dem Format der Textzeilen, die in konventionellen Werkzeugtabellen verwendet werden. Eine abschließende Antwort von "FINI" beendet die Antwort.
4. Das **db_program** tritt dann in eine Ereignis-Warteschleife ein, um Befehle zu empfangen, die anzeigen, dass Werkzeugdaten von LinuxCNC geändert wurden. Werkzeugdaten Änderungen umfassen:
   - a) Laden der Spindel(T*n* M6)/Entladen(T0 M6)
   - b) Änderung der Werkzeugparameter (z. B. G10L1P*n*)
   - c) Werkzeugauswechslungen (M61Q*n*).

Wenn eine Werkzeugdatenänderung auftritt, sendet LinuxCNC einen Befehl an das **db_program**, bestehend aus einem identifizierenden Befehlsbuchstaben, gefolgt von einer vollständigen oder abgekürzten Werkzeugdatenzeile. Das **db_program** muss mit einer Antwort antworten, um den Empfang zu bestätigen. Enthält die Antwort den Text "*NAK*", wird eine Meldung auf stdout ausgegeben, aber die Ausführung wird fortgesetzt. Die "*NAK*"-Meldung bedeutet einen Mangel an Synchronisation zwischen dem **db_program** und LinuxCNC — der begleitende Text sollte einen Hinweis auf die Ursache des Fehlers geben.

Die Befehle für die Änderung von Werkzeugdaten lauten:

- "p" setzt Datenänderungen, die durch G10L1, G10L10, G10L11 G-Codes verursacht werden. Die Werkzeugdatenzeile enthält alle Elemente einer Textzeile der Werkzeugtabelle.
- "l" spindle_load (T*n*M6). Die Werkzeugdatenzeile enthält nur die *T* und *P* Elemente, welche die jeweilige Werkzeug- und Platznummer angeben.
- "*u*" spindle_unload (T0M6). Die Werkzeugdatenzeile enthält nur die "T" und "P" Elemente zur Identifizierung der entsprechenden Werkzeug- und Platznummer.

NOTE Wenn ein NON_RANDOM-Werkzeugwechsler mit [EMCIO]RANDOM_TOOL_CHANGER=0 (Standardeinstellung) angegeben wird, lautet der Befehl spindle_load für T*n*M6 (oder M61Q*n*): *l Tn P0* (Platz 0 ist die Spindel). Der Befehl spindle_unload für T0M6 lautet: *u T0 P0*.

NOTE Wenn ein RANDOM-Werkzeugwechsler mit [EMCIO]RANDOM_TOOL_CHANGER=1 angegeben wird, so wird bei jedem Werkzeugwechsel ein Paar von „spindle_unload/spindle_load“-Befehlen ausgegeben. Das für T*n*M6 (oder M61Q*n*) ausgegebene Befehlspaar lautet *u Tu Pm* gefolgt von *l Tn P0*, wobei *u* das aktuelle Werkzeug ist, das an die Tasche *m* gesendet werden soll, und *n* das neue Werkzeug ist, das in die Spindel geladen werden soll (Tasche ' 0'). Konventionell wird eine Werkzeugnummer von 0 verwendet, um ein leeres Werkzeug anzugeben,

Anwendung

Die Verwendung eines **db_program** ändert nicht die Art und Weise, wie LinuxCNC arbeitet, sondern bietet Unterstützung für neue Datenbankfunktionen für die Werkzeugverwaltung.

Beispielsweise kann eine **db_program**-Datenbankanwendung die Betriebsstunden für alle Werkzeuge verwalten, indem sie jedes Laden/Entladen eines Werkzeugs verfolgt. Eine Maschine könnte dann drei 6 mm Schaftfräser in den Plätzen 111, 112 und 113 haben, wobei die Datenbankanwendung so programmiert ist, dass sie die Werkzeugnummer 110 dem 6 mm Schaftfräser mit den wenigsten Betriebsstunden zuweist. Dann, wenn ein LinuxCNC-Programm fordert Werkzeug 110, die Datenbank würde die entsprechende Tasche auf der Grundlage der Werkzeugnutzung Geschichte.

Werkzeugdatenänderungen, die in LinuxCNC vorgenommen werden (*p*, *u*, *l* Befehle), werden sofort an die **db_program** übertragen, von der erwartet wird, dass sie ihre Quelldaten synchronisiert. Standardmäßig werden LinuxCNC-Anforderungen für Werkzeugdaten (*g*-Befehle) nur beim Start gestellt. Ein Datenbankprogramm kann Werkzeugverwendungsdaten kontinuierlich aktualisieren, so dass langlebige LinuxCNC-Anwendungen von der Aktualisierung der Werkzeugdaten profitieren können, die von der **db_program** bereitgestellt werden. Der G-Code-Befehl **G10L0** kann verwendet werden, um ein Neuladen von Werkzeugdaten (*g*-Befehl) aus G-Code-Programmen oder per MDI anzufordern. Ein Neuladevorgang wird in der Regel auch von einer grafischen Benutzeroberfläche (GUI) bereitgestellt, sodass bedarfsgesteuerte Neuladevorgänge angefordert werden können. Eine Python-GUI-Anwendung kann beispielsweise Folgendes verwenden:

```
#!/usr/bin/env python3
from  linuxcnc import command
```

```
command().load_tool_table()
```

Alternativ kann ein **db_program** seine lokalen Datenänderungen an LinuxCNC weitergeben, indem es den Schnittstellenbefehl load_tool_table() verwendet. Befehle, die Änderungen an LinuxCNC übertragen, können zurückgewiesen werden, wenn der Interpreter läuft. Der Zustand des Interpreters kann vor der Ausgabe eines load_tool_table()-Befehls überprüft werden. Beispiel:

```
#! /usr/bin/env python3
import linuxcnc
s = linuxcnc.stat()
s.poll()
if s.interp_state == linuxcnc.INTERP_IDLE:
    linuxcnc.command().load_tool_table()
else: # defer loading until interp is idle
   ...
```

Wenn die Datenbankanwendung Tools nach der Initialisierung hinzufügt oder entfernt, muss ein Aufruf an tooldb_tools() mit einer aktualisierten user_tools-Liste ausgegeben werden. Die aktualisierte Werkzeugliste wird bei nachfolgenden get-Befehlen oder load_tool_table()-Anforderungen verwendet.

NOTE	Das Entfernen einer Werkzeugnummer sollte nur dann erfolgen, wenn die Werkzeugnummer derzeit nicht in der Spindel geladen ist.

Debug-Umgebungsvariablen

Das Exportieren der Umgebungsvariablen DB_SHOW ermöglicht Ausgaben von LinuxCNC (nach stdout), welche Werkzeugdaten zeigen, die vom **db_program** beim Start und beim nachfolgenden Nachladen von Werkzeugdaten abgerufen werden.

Der Export der Umgebungsvariable DB_DEBUG ermöglicht LinuxCNC Ausgaben (nach stdout) für zusätzliche Debugging-Informationen über die Schnittstellenaktivität.

Beispielprogramm

Ein Beispiel **db_program** (implementiert als Python-Skript) wird mit den Simulationsbeispielen geliefert. Das Programm führt die erforderlichen Operationen aus:

1. Startversion quittieren
2. Empfangen von Anfragen (engl. requests) nach Werkzeugdaten (engl. tool data): *g* (**get**-Befehl)
3. Aktualisierungen der Werkzeugdaten empfangen: *p* (**put**-Befehl)
4. Aktualisierungen der Werkzeugladung empfangen: *l* (Befehl **load_spindle**)
5. Aktualisierungen zum Entladen von Werkzeugen empfangen: *u* (Befehl **unload_spindle**)

Python-Modul tooldb

Das Beispielprogramm verwendet ein von LinuxCNC bereitgestelltes Python-Modul (*tooldb*), das die Low-Level-Details für die Kommunikation und die Versionsprüfung verwaltet. Dieses Modul verwendet

Callback-Funktionen, die vom **db_program** angegeben werden, um auf den Befehl *g* (get) und die Befehle zu reagieren, die Werkzeugdatenänderungen anzeigen (*p*, *l*, *u*).

Das **db_program** verwendet das Modul *tooldb*, indem es den folgenden Python-Code implementiert:

```
user_tools = list(...) # Liste der verfügbaren Werkzeugnummern

def user_get_tool(toolno):
    # Funktion, die auf 'g' (get) Befehle reagiert
    # wird einmal für jede Werkzeugnummer in user_tools aufgerufen
    ...
def user_put_tool(werkzeugnr,params):
    # Funktion als Antwort auf 'p' (put) Befehle
    ...
def user_load_spindle(toolno,params):
    # Funktion, die auf 'l' (put) Befehle reagiert
    ...
def user_unload_spindle(toolno,params):
    # Funktion, die auf 'u' (put) Befehle reagiert
    ...

#-----------------------------------------------------------
# Anfang:
from tooldb import tooldb_tools # bekannte Werkzeuge identifizieren
from tooldb import tooldb_callbacks # Funktionen identifizieren
from tooldb import tooldb_loop # Hauptschleife

tooldb_tools(user_tools)
tooldb_callbacks(user_get_tool,
                 user_put_tool,
                 user_load_spindle,
                 user_unload_spindle,
                )
tooldb_loop()
```

NOTE Die Verwendung von *tooldb* ist nicht erforderlich - sie wird als Demonstration der erforderlichen Schnittstelle und als Bequemlichkeit für die Implementierung von Python-basierten Anwendungen bereitgestellt, die eine Schnittstelle zu einer externen Datenbank haben.

9.11.2. Simulationskonfigurationen

Simulationskonfigurationen über die AXIS GUI:

1. configs/sim/axis/db_demo/**db_ran**.ini (random_toolchanger)
2. configs/sim/axis/db_demo/**db_nonran**.ini (nonrandom_toolchanger)

Jede Sim-Konfiguration simuliert ein **db_programm**, das eine Datenbank mit 10 Werkzeugen mit den Nummern 10—19 implementiert.

Das **db_program** wird durch ein einzelnes Skript (db.py) und symbolische Links darauf für alternative

Verwendungen bereitgestellt: db_ran.py und db_nonran.py. Standardmäßig implementiert das Skript die Funktion random_toolchanger. Nicht-zufällige Toolchanger-Funktionen werden ersetzt, wenn der Link-Name den Text "*nonran*" enthält.

Die Sim-Konfigurationen demonstrieren die Verwendung des Python-Schnittstellenmoduls *tooldb* und implementieren eine einfache Flat-File-Datenbank, welche die Werkzeugzeitnutzung für mehrere Werkzeuge mit gleichen Durchmessern verfolgt. Die Datenbankregeln unterstützen die Auswahl des Werkzeugs mit der geringsten Betriebszeit.

Die Sim-Konfigurationen verwenden eine primäre Aufgabe zu überwachen und zu reagieren, um Werkzeug-Updates von innerhalb LinuxCNC initiiert. Eine periodische Aufgabe aktualisiert die Werkzeugzeitnutzung in regelmäßigen Abständen. Getrennte, gleichzeitige Tasks sind als Threads implementiert, um den Code zu demonstrieren, der erforderlich ist, wenn Änderungen durch das **db_program** initiiert werden, und um Methoden zur Synchronisierung von LinuxCNC-internen Werkzeugdaten zu demonstrieren. Beispiele umfassen:

1. Aktualisierung der Werkzeugparameter
2. Hinzufügen und Entfernen von Werkzeugnummern

Eine wechselseitige Ausschluss (engl. mutual exclusion) -Sperre wird verwendet, um Daten vor Inkonsistenzen zu schützen, die durch Wettlaufbedingungen (engl. und umgangssprachig race condition) zwischen den Aktualisierungen von LinuxCNC-Tooldaten und den Aktualisierungen der Datenbankanwendung entstehen.

Anmerkungen

Wenn ein **db_program** in Verbindung mit einem zufälligen Werkzeugwechsler ([EMCIO]RANDOM_TOOLCHANGER) verwendet wird, unterhält LinuxCNC eine Datei (*db_spindle.tbl* im Konfigurationsverzeichnis), die aus einer einzigen Werkzeug-Tabellenzeile besteht, die das aktuelle Werkzeug in der Spindel identifiziert.

[1] Kinematik: eine Zwei-Wege-Funktion, um aus dem kartesischen Raum in den Gelenkraum zu transformieren.

[2] Wenn die Maschine (z. B. eine Drehmaschine) nur mit den X-, Z- und A-Achsen gemountet ist und die INI-Datei von LinuxCNC nur die Definition dieser 3 Verbindungen enthält, ist die vorherige Behauptung falsch. Weil wir derzeit haben (Gelenk0 = X, Gelenk 1 = Z, Gelenk 2 = A), die davon ausgeht, dass Gelenk 1 = Y. Um dies in LinuxCNC zum Laufen zu bringen, definieren Sie einfach alle Achsen (XYZA), LinuxCNC verwendet dann eine einfache Schleife in HAL für nicht verwendete Y-Achse.

[3] Dieser Unterabschnitt ist einem viel umfangreicheren Artikel entnommen, der unter https://en.wikipedia.org/wiki/PID_controller zu finden ist.

[4] Ziegler, J. G. und Nichols, N. B. (1942), *Optimum Settings for Automatic Controllers*, Transactions of the ASME, DOI 10.1115/1.2899060 und link:https://web.archive.org/web/20170918055307/http://staff.guilan.ac.ir/staff/users/chaibakhsh/fckeditor_repo/file/documents/Optimum%20Settings%20for%20Automatic%20Controllers%20(Ziegler%20and%20Nichols ,%201942).pdf[Das Internet-Archiv

Anwendung

Chapter 10. Benutzerschnittstellen

10.1. AXIS GUI

10.1.1. Einführung

AXIS ist ein grafisches Frontend für LinuxCNC mit Live-Vorschau und Backplot. Es ist in Python geschrieben und verwendet Tk und OpenGL, um seine Benutzeroberfläche anzuzeigen.

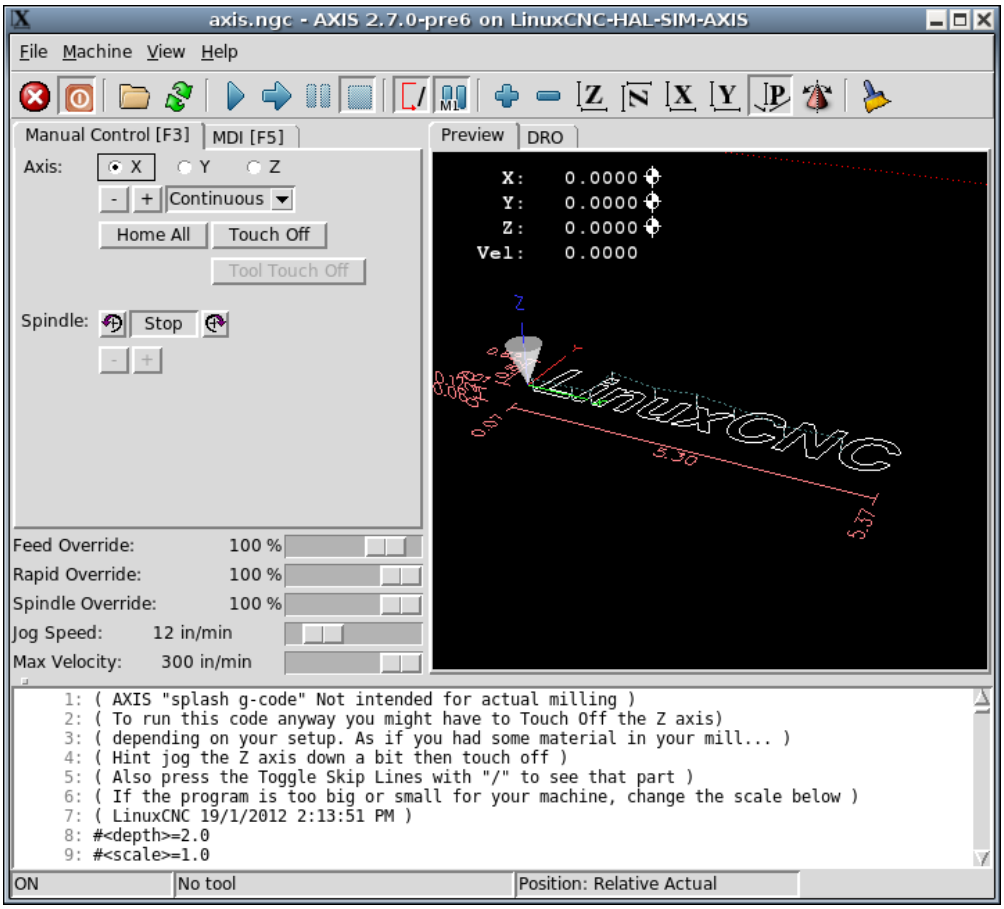


Figure 149. Das AXIS-Fenster

10.1.2. Erste Schritte

Wenn Ihre Konfiguration derzeit nicht für die Verwendung von AXIS eingerichtet ist, können Sie sie ändern, indem Sie die .ini Datei (INI-Datei) bearbeiten. Ändern Sie im Abschnitt *[DISPLAY]* die Zeile *[DISPLAY]* in *DISPLAY = axis*.

Die Beispielkonfiguration "sim/axis.ini" ist bereits für die Verwendung von AXIS als Front-End konfiguriert.

Wenn AXIS gestartet wird, öffnet sich ein Fenster wie das in der Abbildung Das AXIS-Fenster oben.

INI-Einstellungen

Weitere Informationen zur Einstellung der Funktionsweise von AXIS in der INI-Datei, finden Sie in den Abschnitten zu Anzeige und AXIS des Kapitels INI-Konfiguration.

- *CYCLE_TIME* - Passen Sie die Antwortrate der GUI in Millisekunden an. Typisch 100, nutzbarer Bereich 50 - 200
  (akzeptiert Zeit in Sekunden (.05 -.2) aus Legacy-Gründen - Millisekunden bevorzugt, um anderen Bildschirmen zu entsprechen).

```
[DISPLAY]
CYCLE_TIME = 100
```

- PREVIEW_TIMEOUT' - Legt den Timeout für das Laden der G-Code-Vorschau in Sekunden fest. Dauert das Parsen des G-Codes länger als diese Zeitspanne, wird ein Hinweis angezeigt und nur der erste Teil des Programms wird in der grafischen Anzeige dargestellt. Die Angabe von 0 oder das Weglassen der Einstellung führt zu keinem Timeout.

```
[DISPLAY]
PREVIEW_TIMEOUT = 5
```

Eine typische Sitzung

1. Start von LinuxCNC und Auswahl einer Konfigurationsdatei.
2. Geben Sie den Notaus frei (F1) und schalten Sie das Gerät ein (F2).
3. Referenzfahrt aller Achsen.
4. Laden der G-Code-Datei.
5. Verwenden Sie das Vorschaudiagramm, um zu überprüfen, ob das Programm korrekt ist.
6. Laden des Materials.
7. Stellen Sie den richtigen Versatz für jede Achse ein, indem Sie joggen und bei Bedarf die Taste "Touch Off" verwenden.
8. Führen Sie das Programm aus.
9. Um dieselbe Datei erneut zu bearbeiten, kehren Sie zu Schritt 6 zurück. Um eine andere Datei zu bearbeiten, kehren Sie zu Schritt 4 zurück.
10. Wenn der Auftrag abgeschlossen ist, beenden Sie AXIS.

NOTE

Nun notwendige Schritte um dasselbe Programm erneut auszuführen, hängen von Ihrem Setup und Ihren Anforderungen ab. Möglicherweise müssen Sie mehr Material laden und Offsets setzen oder einen Offset verschieben und festlegen und dann das Programm erneut ausführen. Wenn Ihr Material fixiert ist, müssen Sie das Programm möglicherweise nur erneut ausführen. Weitere Informationen zum Befehl run finden Sie im Abschnitt zum Menü Maschine.

10.1.3. AXIS Fenster

Das AXIS-Fenster enthält die folgenden Elemente:

- Ein Anzeigebereich, der Folgendes anzeigt:
  - Eine Vorschau der geladenen Datei (in diesem Fall *axis.ngc*) sowie des aktuellen Speicherorts des "kontrollierten Punktes" der CNC-Maschine. Später wird in diesem Bereich der Pfad angezeigt, den die CNC-Maschine durchlaufen hat, der als "Backplot" bezeichnet wird.
  - Eine große Anzeige der aktuellen Position und aller Offsets.
- Eine Menüleiste und Symbolleiste, mit der Sie verschiedene Aktionen ausführen können
- *Manuelle Steuerungsregisterkarte* (engl. Manual Control Tab) - mit der Sie die Maschine bewegen können, die Spindel ein- oder ausschalten und das Kühlmittel ein- oder ausschalten können, wenn es in der INI-Datei enthalten ist.
- *MDI Tab* - in dem G-Code-Programme manuell eingegeben werden können, eine Zeile nach der anderen. Dies zeigt auch die *Aktive G-Codes*, die zeigen, welche modalen G-Codes in Kraft sind.
- *Vorschub Neufestsetzung* (engl. feed override) - mit dem Sie die Geschwindigkeit programmierter Bewegungen skalieren können. Das Standardmaximum beträgt 120 % und kann in der INI-Datei auf einen anderen Wert festgelegt werden. Weitere Informationen finden Sie im Abschnitt Anzeige der INI-Datei.
- *Spindel Neufestsetzung* (engl. spindle override) - mit dem Sie die Spindelgeschwindigkeit nach oben oder unten skalieren können.
- *Jog Speed* - mit dem Sie die Jog-Geschwindigkeit innerhalb der in der INI-Datei festgelegten Grenzen einstellen können. Weitere Informationen finden Sie im Abschnitt Anzeige der INI-Datei.
- *Max. Geschwindigkeit* (engl. max velocity) - ermöglicht es Ihnen, die maximale Geschwindigkeit aller programmierten Bewegungen zu begrenzen (außer spindelsynchronisierte Bewegung).
- Ein Textanzeigebereich, der den geladenen G-Code anzeigt.
- A status bar which shows the state of the machine. In this screen shot, the machine is turned on, does not have a tool inserted, and the displayed position is *Relative* (showing all offsets), and *Actual* (showing feedback position). Its last field is the loaded program's estimated *Run time*, which while the program runs becomes the time so far beside the run it is heading for; it is empty when the program could not be timed. See *G-code properties* below for what the estimate does and does not model.

Menüpunkte

Einige Menüelemente sind möglicherweise ausgegraut, je nachdem, wie Sie Ihre INI-Datei konfiguriert haben. Weitere Informationen zur Konfiguration finden Sie im Kapitel INI.

Menü "Datei"

- *Öffnen...* - Öffnet ein Standarddialogfeld zum Öffnen einer G-Code-Datei, die in AXIS geladen werden soll. Wenn Sie LinuxCNC für die Verwendung eines Filterprogramms konfiguriert haben, können Sie es auch öffnen. Weitere Informationen finden Sie im Abschnitt FILTER der INI-Konfiguration.

- „Letzte Dateien“ – Zeigt eine Liste der zuletzt geöffneten Dateien an.
- *Bearbeiten...* - Öffnen Sie die aktuelle G-Code-Datei zur Bearbeitung, wenn Sie einen Editor in Ihrer INI-Datei konfiguriert haben. Weitere Informationen zum Angeben eines zu verwendenden Editors finden Sie im Abschnitt DISPLAY.
- *Reload* - Laden Sie die aktuelle G-Code-Datei neu. Wenn Sie es bearbeitet haben, müssen Sie es neu laden, damit die Änderungen wirksam werden. Wenn Sie eine Datei stoppen und von vorne beginnen möchten, laden Sie die Datei neu. Das Neuladen der Symbolleiste ist identisch mit dem Menü.
- *G-Code speichern unter...* - Speichern Sie die aktuelle Datei unter einem neuen Namen.
- *G-code properties* - The program's size, its rapid and feed path lengths, its bounds, and its run time. The run time is an estimate made while the program was parsed: it models the machine's velocity and acceleration limits, the corners between moves, the feed modes (G93, G95, G96), dwells and tool changes, and it assumes 100% feed, rapid and spindle override. It is not a trajectory planner, so expect a few percent either way; and it does not model overrides, adaptive feed, M0/M1/M66 waits or spindle spin-up, so a run that waits takes longer than it says. It reads `[DISPLAY]TOOL_CHANGE_SECONDS` for what one M6 costs, and shows *not available* on a machine whose INI states no velocity limits at all.
- *Werkzeugtabelle bearbeiten...* - Wie bei Bearbeiten, wenn Sie einen Editor definiert haben, können Sie die Werkzeugtabelle öffnen und bearbeiten.
- *Werkzeugtabelle neu laden* - Nach dem Bearbeiten der Werkzeugtabelle müssen Sie diese neu laden.
- *Ladder Editor* - Wenn Sie ClassicLadder geladen haben, können Sie es von hier aus bearbeiten. Weitere Informationen finden Sie im Kapitel ClassicLadder.
- *Beenden* - Beendet die aktuelle LinuxCNC-Sitzung.

Maschinen-Menü

- *Notaus F1 ein-/ausschalten* - Ändern Sie den Zustand des Notaus.
- *Toggle Machine Power F2* - Ändern Sie den Zustand der Maschinenleistung, wenn der Notaus nicht eingeschaltet ist.
- *Programm ausführen* - Führt das aktuell geladene Programm von Anfang an aus.
- „Ab ausgewählter Zeile ausführen“ – Wählen Sie die Zeile aus, bei der Sie zuerst beginnen möchten. Seien Sie vorsichtig, da dies das Werkzeug zuerst an die erwartete Position vor der Zeile bewegt und dann den Rest des Codes ausführt.

WARNING	Verwenden Sie nicht *Run From Selected Line*, wenn Ihr G-Code-Programm Unterroutinen enthält.

- *Step* - Einzelner Schritt durch ein Programm.
- Pause - Unterbrechung eines Programms.
- *Fortsetzen* (engl. resume) - Wiederaufnahme der Ausführung nach einer Pause.
- *Stop* - Stoppt ein laufendes Programm. Folgt nach einem Stopp erneut das Kommando "Ausführen!", so wird das Programm von vorne gestartet.

- *Stop at M1* - Wenn ein M1 erreicht ist und dies aktiviert ist, wird die Programmausführung auf der M1-Leitung angehalten. Drücken Sie Fortsetzen, um fortzufahren.
- *Zeilen mit "/" überspringen* - Wenn eine Zeile mit / beginnt und dies aktiviert ist, wird die Zeile übersprungen.
- *MDI-Verlauf löschen* - Löscht das MDI-Verlaufsfenster.
- *Aus MDI-Verlauf kopieren* - Kopiert den MDI-Verlauf in die Zwischenablage
- *In MDI-Verlauf einfügen* - Einfügen aus der Zwischenablage in das MDI-Verlaufsfenster
- *Kalibrierung* - Startet den Kalibrierungsassistenten (emccalib.tcl). Die Kalibrierung liest die HAL-Datei und erstellt für jedes *setp*, das eine Variable aus der INI-Datei verwendet, die sich in einem [AXIS_L],[JOINT_N],[SPINDLE_S] oder [TUNE] Abschnitt befindet, ein Eintrag, der bearbeitet und getestet werden kann.
- *HAL-Konfiguration anzeigen* - Öffnet das Fenster HAL-Konfiguration, in dem Sie HAL-Komponenten, Pins, Parameter, Signale, Funktionen und Threads überwachen können.
- *HAL-Messgerät* - Öffnet ein Fenster, in dem Sie einen einzelnen HAL-Pin, ein Signal oder einen Parameter überwachen können.
- *HAL Scope* - Öffnet ein virtuelles Oszilloskop zur Anzeige von HAL-Werten (vertikal) über die Zeit (horizontal) ermöglicht.
- *LinuxCNC-Status anzeigen* - Öffnet ein Fenster mit dem Status von LinuxCNC.
- *Debug Level festlegen* - Öffnet ein Fenster, in dem Debug-Ebenen angezeigt und einige festgelegt werden können.
- *Referenzfahrt* (engl. Homing) - Eine oder alle Achsen zu Referenzpunkt führen.
- *Unhoming* - Unhoming einer oder aller Achsen.
- *Nullkoordinatensystem* - Setzt alle Versätze im gewählten Koordinatensystem auf Null.
- Werkzeug Touch Off
  - *Werkzeug auf Werkstück aufsetzen* - Beim Ausführen von Touch Off bezieht sich der eingegebene Wert auf das aktuelle Werkstückkoordinatensystem (*G5x*), modifiziert durch den Achsversatz (*G92*). Wenn die Berührung abgeschlossen ist, wird die relative Koordinate für die gewählte Achse zum eingegebenen Wert. Siehe G10 L10 im G-Code-Kapitel.
  - *Werkzeug-Touch Off to Fixture* - Beim Ausführen von Touch Off ist der eingegebene Wert relativ zum neunten (*G59.3*) Koordinatensystem, wobei der Achsenversatz (*G92*) ignoriert wird. Dies ist nützlich, wenn an einer festen Position auf der Maschine eine Werkzeug-Berührungshalterung vorhanden ist, wobei das neunte (*G59.3*) Koordinatensystem so eingestellt ist, dass sich die Spitze eines Werkzeugs mit der Länge Null am Ursprung der Halterung befindet, wenn die Relative Koordinaten sind 0. Siehe G10 L11 im G-Code-Kapitel.

Ansicht-Menü

- *Draufsicht* (engl. top view) – Die Draufsicht (oder Z-Ansicht) zeigt den G-Code entlang der Z-Achse von positiv nach negativ. Diese Ansicht eignet sich am besten zum Betrachten von X und Y.
- *Gedrehte Draufsicht* (engl. rotated top view) - Die gedrehte Draufsicht (oder gedrehte Z-Ansicht) zeigt auch den G-Code an, der entlang der Z-Achse von positiv nach negativ aussieht. Aber manchmal ist

es praktisch, die X & Y-Achsen um 90 Grad gedreht anzuzeigen, um besser zum Display zu passen. Diese Ansicht eignet sich auch bestens, um X & Y zu betrachten.

- *Seitenansicht* (engl. side view) - Die Seitenansicht (oder X-Ansicht) zeigt den G-Code an, der entlang der X-Achse von positiv nach negativ aussieht. Diese Ansicht eignet sich am besten für den Blick auf Y & Z.
- *Vorderansicht* - Die Vorderansicht (oder Y-Ansicht) zeigt den G-Code an, der entlang der Y-Achse von negativ nach positiv aussieht. Diese Ansicht eignet sich am besten für den Blick auf X & Z.
- *Perspektivische Ansicht* (engl. perspective view) - Die perspektivische Ansicht (oder P-Ansicht) zeigt den G-Code an, der das Teil aus einem einstellbaren Blickwinkel betrachtet, standardmäßig X+, Y-, Z+. Die Position ist mit der Maus und dem Zug-/Drehwahlschalter einstellbar. Diese Ansicht ist eine Kompromissansicht, und obwohl sie versucht, drei (bis neun!) Diese Ansicht ist am besten, wenn Sie alle drei (bis neun) Achsen gleichzeitig sehen möchten.

Sichtweise

Das AXIS-Anzeigeauswahlmenü "Ansicht" bezieht sich auf die Ansichten "Oben", "Vorne" und "Seitlich". Diese Begriffe sind korrekt, wenn die Z-Achse der CNC-Maschine senkrecht steht, mit positivem Z nach oben. Dies gilt für vertikale Fräsmaschinen, was wahrscheinlich die häufigste Anwendung ist, und auch für fast alle Erodiermaschinen und sogar vertikale Revolverdrehbänke, bei denen sich das Teil unter dem Werkzeug dreht.

Die Begriffe *Oben* (engl. top), *Vorne* (engl. front) und *Seitlich* (engl. side) können verwirrend sein bei anderen CNC-Maschinen, wie z.B. bei einer Standard-Drehmaschine, bei der die Z-Achse horizontal verläuft, oder bei einer horizontalen Fräsmaschine, bei der die Z-Achse ebenfalls horizontal verläuft, oder sogar bei einer umgekehrten vertikalen Revolverdrehmaschine, bei der sich das Werkstück über dem Werkzeug dreht und die positive Richtung der Z-Achse nach unten verläuft!

Denken Sie nur daran, dass die positive Z-Achse (fast) immer vom Werkstück entfernt ist. Seien Sie also mit der Konstruktion Ihrer Maschine vertraut und interpretieren Sie die Anzeige nach Bedarf.

- *Display Inches* - Legt die AXIS-Anzeigeskalierung für Zoll fest.
- *Display MM* - Legt die AXIS Display-Skalierung auf Millimeter fest.
- *Programm anzeigen* - Die Vorschauanzeige des geladenen G-Code-Programms kann auf Wunsch vollständig deaktiviert werden.
- *Zeige Vorschau von Eilgängen* (engl. show program rapids) - Die Vorschauanzeige des geladenen G-Code-Programms zeigt die Vorschubbewegungen (G1,G2,G3) immer in weiß an. Die Anzeige von Eilgängen (G0) in cyan kann jedoch auf Wunsch deaktiviert werden.
- *Alpha-Blend-Programm* - Diese Option macht die Vorschau komplexer Programme leichter sichtbar, kann aber dazu führen, dass die Vorschau langsamer angezeigt wird.
- *Show Live Plot* - Die Hervorhebung der Vorschubpfade (G1,G2,G3) während der Bewegungen des Werkzeugs kann auf Wunsch deaktiviert werden.

- *Werkzeug anzeigen* - Die Anzeige des Werkzeugkegels/-zylinders kann auf Wunsch deaktiviert werden.
- *Zeige Ausdehnung* - Die Anzeige der Ausdehnung (engl. extents) (maximaler Verfahrweg in jeder Achsenrichtung) des geladenen G-Code-Programms kann auf Wunsch deaktiviert werden.
- *Offsets anzeigen* – Der ausgewählte Fixture Offset (G54-G59.3) Ursprungsort kann als Satz von drei orthogonalen Linien angezeigt werden, jeweils eine aus rot, blau und grün. Diese Offset-Ursprungsanzeige (oder Fixture Zero) kann auf Wunsch deaktiviert werden.
- *Maschinenlimits anzeigen* - Die maximalen Verfahrwege der Maschine für jede Achse, wie sie in der INI-Datei festgelegt sind, werden als rechteckiges Feld in roten, gestrichelten Linien dargestellt. Dies ist nützlich, wenn Sie ein neues G-Code-Programm laden oder prüfen, wie viel Fixture-Offset benötigt wird, um das G-Code-Programm innerhalb der Reisegrenzen Ihrer Maschine zu bringen. Es kann abgeschaltet werden, wenn es nicht benötigt wird.
- *Geschwindigkeit anzeigen* - Eine Anzeige der Geschwindigkeit ist manchmal nützlich, um zu sehen, wie nah Ihre Maschine an ihren Entwurfsgeschwindigkeiten läuft. Sie kann auf Wunsch deaktiviert werden.
- *Restweg anzeigen* (engl. Show Distance to Go) - Der Restweg ist ein sehr nützlicher Hinweis, wenn Sie ein unbekanntes G-Code-Programm zum ersten Mal ausführen. In Kombination mit den Eilgang- und Vorschub-Override-Steuerungen können unerwünschte Werkzeug- und Maschinenschäden vermieden werden. Sobald das G-Code-Programm fehlerfrei läuft, kann die Restweg-Anzeige auf Wunsch deaktiviert werden.
- *Koordinaten in großer Schrift...* - Die Koordinaten der Achsen und die Geschwindigkeit im Voraus werden in großer Schrift in der Werkzeugwegansicht angezeigt.
- *Live Plot löschen* - Während das Werkzeug in der AXIS-Anzeige reist, wird der G-Code-Pfad hervorgehoben. Um das Programm zu wiederholen oder einen Interessenbereich besser zu sehen, können die zuvor markierten Pfade gelöscht werden.
- *Zeige befohlene Position* (engl. show commanded position) - Dies ist die Position, die LinuxCNC versuchen wird zu gehen. Sobald die Bewegung gestoppt wurde, ist dies die Position, die LinuxCNC versuchen wird, zu halten.
- *Ist-Position anzeigen* (engl. show actual position) - Die Ist-Position ist die gemessene Position, wie sie von den Encodern des Systems zurückgelesen oder von Schrittgeneratoren simuliert wird. Diese kann aus vielen Gründen, wie z. B. PID-Abstimmung, physikalische Einschränkungen oder Positionsquantisierung, leicht von der befohlenen Position abweichen.
- *Maschinenposition anzeigen* (engl. show machine position) - Dies ist die Position in nicht verschobenen Koordinaten, wie sie bei der Referenzfahrt ermittelt wurde.
- *Relative Position anzeigen* (engl. show relative position) - Dies ist die Maschinenposition, modifiziert durch die Offsets "G5x", "G92" und "G43".

Hilfemenü

- "Über AXIS" - Wir alle wissen, was das ist.
- *Kurzübersicht* - Zeigt die Tastenkombinationen an.

Schaltflächen der Symbolleiste

Von links nach rechts in der AXIS-Anzeige lauten die Schaltflächen der Symbolleiste (Tastenkombinationen werden [in Klammern] angezeigt):

- Umschalten des Notausschalters [F1] (auch E-Stop genannt)
- Umschalten Machinenstrom [F2]
- G-Code-Datei öffnen [O]
- Aktuelle Datei neu laden [Strg-R]
- Beginn der Ausführung der aktuellen Datei [R]
- Nächste Zeile ausführen [T]
- Ausführung anhalten [P] Ausführung fortsetzen [S]
- Programmausführung anhalten [ESC]
- Zeilen überspringen mit "/" [Alt-M-/] umschalten
- Optionale Pause einschalten [Alt-M-1]
- Vergrößern (engl. zoom in)
- Herauszoomen (engl. zoom out)
- Draufsicht
- Gedrehte Draufsicht
- Seitenansicht
- Vorderansicht
- Perspektivische Ansicht
- Umschalten zwischen Ziehen und Drehen [D]
- Live-Backplot löschen [Strg-K]

Grafischer Anzeigebereich

Koordinatenanzeige

In der oberen linken Ecke der Programmanzeige befindet sich die Anzeige der Koordinatenposition für jede Achse. Rechts neben der Nummer wird ein Ursprungssymbol angezeigt, wenn die Achse referenziert wurde.

Ein Grenzwertsymbol wird rechts neben der Koordinatenpositionsnummer angezeigt, wenn die Achse an einem ihrer Endschalter steht.

Um die Positionsnummern richtig zu interpretieren, beachten Sie die Anzeige "Position:" in der Statusleiste. Wenn die Position "Maschinen-Ist" lautet, dann ist die angezeigte Zahl im Maschinenkoordinatensystem. Steht sie auf "Relativ Aktuell", dann ist die angezeigte Zahl im Offset-Koordinatensystem. Wenn die angezeigten Koordinaten relativ sind und ein Offset eingestellt wurde, enthält die Anzeige eine cyanfarbene Markierung *Maschinen-Ursprung* (engl. machine origin) .

Ist die Position *Befohlen* (engl. commanded), wird die genaue Koordinate angezeigt, die in einem G-Code-Befehl angegeben wurde. Ist die Position *Ist* (engl. actual), dann ist es von der Maschine tatsächlich angefahrene Position. Diese Werte können aufgrund von Schleppfehler, Totzone, Messgeräteauflösung oder Schrittweite von der befohlenen Position abweichen. Wenn Sie z. B. eine Bewegung mit X 0,0033 auf Ihrer Fräsmaschine befehlen, aber ein Schritt Ihres Schrittmotors oder eine Encoderzählung 0,00125 beträgt, dann könnte die *befohlene* Position 0,0033 sein, aber die *tatsächliche* Position wird 0,0025 (2 Schritte) oder 0,00375 (3 Schritte) sein.

Vorschau-Plot

Wird eine Datei geladen, so wird im Anzeigebereich eine Vorschau angezeigt. Schnelle Bewegungen (z.B. durch den Befehl *G0*) werden als cyanfarbene Linien dargestellt. Bewegungen im Vorschub (z. B. mit dem Befehl "G1") werden als durchgezogene weiße Linien dargestellt. Verweilzeiten (z. B. durch den Befehl "G4") werden als kleine rosa "X"-Markierungen dargestellt.

G0 (Eilgang) Bewegungen vor einer Vorschubbewegung werden nicht in der Vorschau angezeigt. Eilgangbewegungen nach einem T<n> (Werkzeugwechsel) werden erst nach der ersten Vorschubbewegung in der Vorschau angezeigt. Um eine dieser Funktionen auszuschalten, programmieren Sie einen G1 ohne Bewegungen vor den G0-Bewegungen.

Programm-Extents

Die *Ausdehnungen* des Programms in jeder Achse werden angezeigt. An den Enden werden die kleinsten und größten Koordinatenwerte angegeben. In der Mitte ist die Differenz zwischen den Koordinaten dargestellt.

Wenn einige Koordinaten die "weichen Grenzen" in der INI-Datei überschreiten, wird die betreffende Abmessung in einer anderen Farbe angezeigt und von einem Kästchen umgeben. In der nachstehenden Abbildung ist die maximale weiche Grenze auf der X-Achse überschritten, was durch das Kästchen um den Koordinatenwert angezeigt wird. Der minimale X-Verfahrweg des Programms ist -1,95, der maximale X-Verfahrweg ist 1,88, und das Programm benötigt einen X-Verfahrweg von 3,83 Zoll. Um das Programm so zu verschieben, dass es sich innerhalb des Verfahrwegs der Maschine befindet, gehen Sie nach links und berühren Sie die X-Position erneut.



Figure 150. Weiche Grenzwerte (engl. soft limits)

Werkzeugkegel

Wenn kein Werkzeug geladen ist, wird die Position der Werkzeugspitze durch den „Werkzeugkegel“ angezeigt. Der „Werkzeugkegel“ gibt keine Auskunft über Form, Länge oder Radius des Werkzeugs.

Wenn ein Werkzeug geladen wird (z.B. mit dem MDI-Befehl *T1 M6*), ändert sich der Kegel in einen Zylinder, der den in der Werkzeugtabellendatei angegebenen Durchmesser des Werkzeugs anzeigt.

Backplot

Wenn sich die Maschine bewegt, hinterlässt sie eine Spur, den so genannten Backplot. Die Farbe der Linie gibt die Art der Bewegung an: Gelb für Jogging, blassgrün für schnelle Bewegungen, rot für gerade Bewegungen mit Vorschubgeschwindigkeit und magenta für kreisförmige Bewegungen mit Vorschubgeschwindigkeit.

Raster

AXIS kann in orthogonalen Ansichten optional ein Raster anzeigen. Aktivieren oder deaktivieren Sie das Raster über das Menü "Raster" unter "Ansicht". Wenn es aktiviert ist, wird das Gitter in der Draufsicht und der gedrehten Draufsicht angezeigt; wenn das Koordinatensystem nicht gedreht ist, wird das Gitter auch in der Vorder- und Seitenansicht angezeigt. Die Voreinstellungen im Menü *Raster* werden durch den Eintrag `[DISPLAY]GRIDS` in der INI-Datei gesteuert. Wenn nichts angegeben wird, ist die Voreinstellung `10mm 20mm 50mm 100mm 1in 2in 5in 10in`.

Die Angabe eines sehr kleinen Rasters kann die Leistung verringern.

Interaktionen

Wenn Sie mit der linken Maustaste auf einen Teil des Vorschaudiagramms klicken, wird die Linie sowohl in der grafischen Darstellung als auch in der Textanzeige hervorgehoben. Wenn Sie mit der linken Maustaste auf einen leeren Bereich klicken, wird die Hervorhebung wieder entfernt.

Durch Ziehen mit gedrückter linker Maustaste wird die Vorschaudarstellung verschoben (Panning).

Durch Ziehen mit gedrückter linker Maustaste bei gedrückter Umschalttaste oder durch Ziehen mit gedrücktem Mausrad wird das Vorschaubild gedreht. Bei einer hervorgehoben Linie ist der Drehpunkt der Mittelpunkt der Linie. Andernfalls ist der Drehpunkt der Mittelpunkt des gesamten Programms.

Durch Drehen des Mausrads oder durch Ziehen mit gedrückter rechter Maustaste oder durch Ziehen mit der Steuerung und gedrückter linker Maustaste wird die Vorschaudarstellung vergrößert oder verkleinert.

Durch Anklicken eines der Symbole "Voreingestellte Ansicht" oder durch Drücken von "V" können mehrere voreingestellte Ansichten ausgewählt werden.

Textanzeigebereich

Wenn Sie mit der linken Maustaste auf eine Zeile des Programms klicken, wird diese Zeile sowohl in der grafischen als auch in der Textanzeige hervorgehoben.

Wenn das Programm läuft, wird die Zeile, die gerade ausgeführt wird, rot hervorgehoben. Wenn der Benutzer keine Zeile ausgewählt hat, wird die Textanzeige automatisch umgeschaltet, um die aktuelle Zeile anzuzeigen.

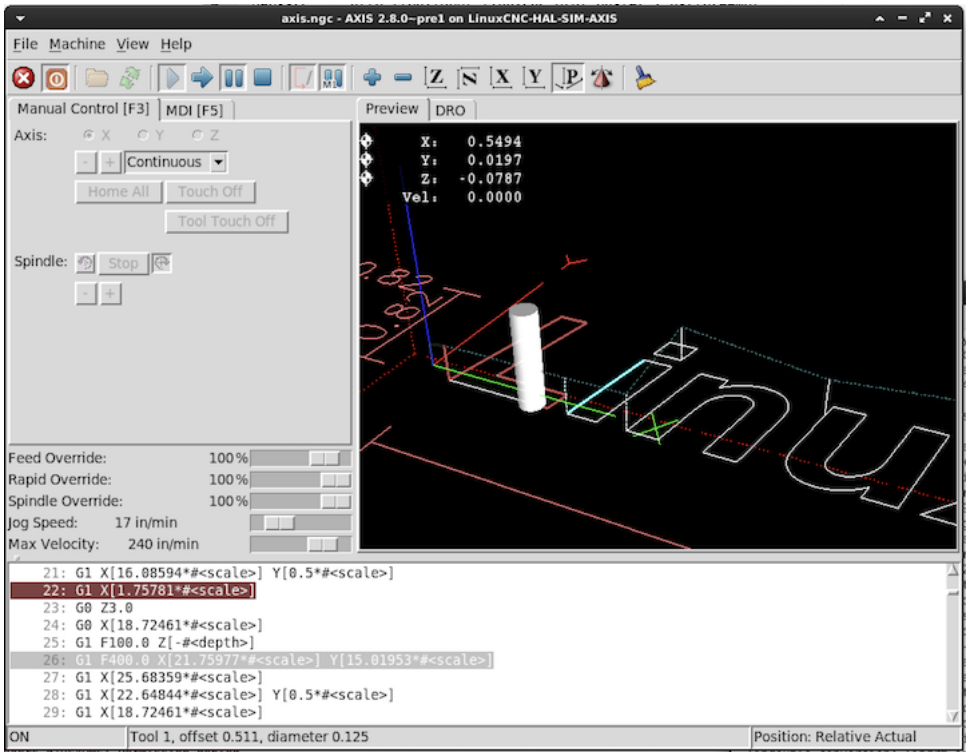


Figure 151. Aktuelle und ausgewählte Zeilen

Manuelle Steuerung

Wenn die Maschine eingeschaltet ist, aber kein Programm abläuft, können die Elemente auf der Registerkarte "Manuelle Steuerung" verwendet werden, um die Maschine zu bewegen oder ihre Spindel und Kühlmittel zu steuern.

Wenn das Gerät nicht eingeschaltet ist oder wenn ein Programm läuft, sind die manuellen Bedienelemente nicht verfügbar.

Viele der im Folgenden beschriebenen Elemente sind nicht bei allen Maschinen sinnvoll. Wenn AXIS feststellt, dass ein bestimmter Pin in HAL nicht angeschlossen ist, wird das entsprechende Element auf der Registerkarte "Manuelle Steuerung" entfernt. Ist zum Beispiel der HAL-Pin "spindle.0.brake" nicht angeschlossen, erscheint die Schaltfläche "Brake" nicht auf dem Bildschirm. Ist die Umgebungsvariable *AXIS_NO_AUTOCONFIGURE* gesetzt, so ist dieses Verhalten deaktiviert und alle Elemente werden angezeigt.

Die Achsengruppe

Mit *AXIS* können Sie die Maschine manuell bewegen. Diese Aktion wird als "Jogging" bezeichnet. Wählen Sie zunächst die zu bewegende Achse durch Anklicken aus. Klicken Sie dann auf die Schaltfläche "+" oder "-" und halten Sie sie gedrückt, je nachdem, in welche Richtung Sie verfahren möchten. Die ersten vier Achsen können auch mit den Pfeiltasten (X und Y), den Tasten PAGE UP und PAGE DOWN (Z) und den Tasten [und] (A) bewegt werden.

Wenn Sie "Kontinuierlich" auswählen, wird die Bewegung so lange fortgesetzt, wie die Schaltfläche oder Taste gedrückt wird. Wenn ein anderer Wert gewählt wird, bewegt sich die Maschine bei jedem Klicken auf die Schaltfläche oder Drücken der Taste genau um die angezeigte Strecke. Standardmäßig sind die folgenden Werte verfügbar: "0.1000, 0.0100, 0.0010, 0.0001".

Siehe den DISPLAY Abschnitt für weitere Informationen zum Einstellen der Schrittweiten.

Referenzfahrt (engl. homing) (Identitätskinematik)

Die INI-Datei Einstellung [KINS]JOINTS definiert die Gesamtzahl der Gelenke für das System. Ein Gelenk kann mit einem Referenzpunktschalter oder für eine "sofortige" Referenzfahrt konfiguriert werden. Gelenke können eine Referenzfahrt-Reihenfolge angeben, um die Reihenfolge der Referenzfahrt für Gruppen von Gelenken zu organisieren.

Wenn **alle** Gelenke für die Referenzfahrt konfiguriert sind und über gültige Referenzfahrten verfügen, zeigt die Referenzfahrt-Schaltfläche "Alle Referenzfahrten" an. Durch Drücken der Schaltfläche "Alle referenzieren" (oder der Taste Strg-Pos1 (engl. Ctrl-HOME)) wird die Referenzfahrt für alle Gelenke unter Verwendung ihrer definierten Referenzfahrt-Sequenzen eingeleitet. Durch Drücken der Taste Pos1/HOME wird die Referenzfahrt für das Gelenk, das der aktuell ausgewählten Achse entspricht, eingeleitet, auch wenn keine Referenzfahrtsequenz definiert ist.

Wenn nicht alle Achsen über gültige Referenzfahrt-Sequenzen verfügen, zeigt die Referenzfahrt-Schaltfläche "Home Axis" (Referenzfahrt-Achse) an und führt die Referenzfahrt nur für die aktuell ausgewählte Achse durch. Jede Achse muss separat ausgewählt und referenziert werden.

Das Dropdown-Menü Maschine/Referenzierung bietet eine alternative Methode zum Referenzieren von Achsen. Das Dropdown-Menü Maschine/Unhoming bietet die Möglichkeit, die Referenzfahrt von Achsen aufzuheben.

If your machine does not have home switches defined in the configuration, the *Home* button will set the current position of the selected axis as the absolute position 0 for that axis and will set the *is-homed* bool for that axis.

Weitere Informationen finden Sie im Kapitel Referenzfahrt Konfiguration.

Referenzfahrt (engl. homing) (nicht-Identität-Kinematik)

Die Bedienung ist ähnlich wie bei der Identitätskinematik, aber vor der Referenzfahrt wählen die Auswahlknöpfe die Gelenke nach Nummern aus. Die Schaltfläche für die Referenzfahrt zeigt "Home All" an, wenn alle Gelenke für die Referenzfahrt konfiguriert sind und über gültige Referenzfahrten verfügen. Andernfalls zeigt die Schaltfläche für die Referenzfahrt "Home Joint" an.

Weitere Informationen finden Sie im Kapitel Referenzfahrt Konfiguration.

Touch-Off

Durch Drücken von *Touch Off* oder der END-Taste wird der *G5x-Offset* für die aktuelle Achse geändert, so dass der aktuelle Achsenwert dem angegebenen Wert entspricht. Ausdrücke können nach den Regeln für rs274ngc-Programme eingegeben werden, mit der Ausnahme, dass auf Variablen nicht Bezug genommen werden darf. Der resultierende Wert wird als Zahl angezeigt.

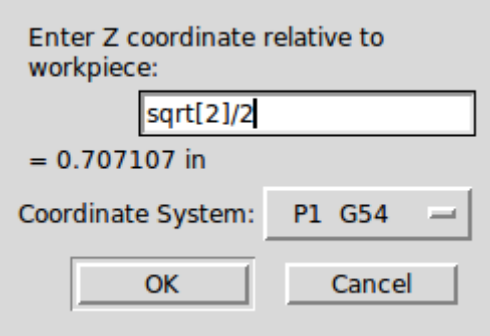


Figure 152. Touch Off Fenster

Siehe auch die Optionen im Menü Maschine: "Werkstück berühren" und "Werkstückhalter berühren".

Actual Position (tatsächliche Position) Touch Off

In der .INI-Datei kann für eine Achse eingestellt werden, dass ihr Ist-Positionswert in die Antast-Berechnung einbezogen wird, wahlweise addierend oder subtrahierend. Dies ist vor allem für Maschinen mit einer nicht motorisierten Achse (z. B. Pinole mit Encoder) nützlich. Wenn diese Funktion für eine Achse aktiviert ist, zeigt die Titelleiste des Antast-Fensters **(system ACTUAL)** an.

Siehe Touch Off mit der Actual Position (tatsächliche Position) für weitere Informationen.

Werkzeug Touch Off

Durch Drücken der Schaltfläche *Tool Touch Off* werden die Werkzeuglänge und die Offsets des aktuell geladenen Werkzeugs so verändert, dass die aktuelle Position der Werkzeugspitze mit der eingegebenen Koordinate übereinstimmt.

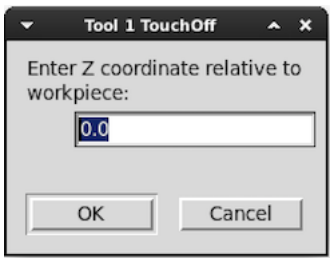


Figure 153. Werkzeug Touch Off Fenster

Siehe auch die Optionen *Werkzeug berühren auf Werkstück* und *Werkzeug berühren auf Halterung* im Menü Maschine.

Grenzwerte überschreiten

Wenn Sie auf "Grenzen außer Kraft setzen" (engl. override limits) klicken, kann die Maschine vorübergehend über einen physischen Endschalter hinausfahren. Dieses Kontrollkästchen ist nur verfügbar, wenn ein Endschalter ausgelöst wird. Die Überbrückung wird nach einem Tippen zurückgesetzt. Wenn die Achse mit separaten positiven und negativen Endschaltern konfiguriert ist, wird LinuxCNC das Joggen nur in der richtigen Richtung erlauben. Die Funktion *Override Limits erlaubt kein Überschreiten eines Softlimits. Der einzige Weg, um eine weiche Grenze auf einer Achse zu deaktivieren ist, um einen neuen Referenzpunkt zu bestimmen (engl. unhome).*

Die Spindel-Gruppe

Mit den Buttons in der ersten Reihe wählen Sie die Drehrichtung der Spindel aus: Gegen den

Uhrzeigersinn, Angehalten, Im Uhrzeigersinn. Gegen den Uhrzeigersinn wird nur angezeigt, wenn der Pin *spindle.0.reverse* in der HAL-Datei enthalten ist (er kann *net trick-axis spindle.0.reverse* sein). Die Schaltflächen in der nächsten Zeile erhöhen oder verringern die Drehgeschwindigkeit. Mit dem Kontrollkästchen in der dritten Zeile kann die Spindelbremse aktiviert oder deaktiviert werden. Je nach Maschinenkonfiguration werden möglicherweise nicht alle Elemente in dieser Gruppe angezeigt. Durch Drücken der Spindelstarttaste wird die *S*-Drehzahl auf 1 gesetzt.

Die Kühlmittelgruppe

Mit den beiden Schaltflächen können die Kühlmittel *Nebel* und *Flut* ein- und ausgeschaltet werden. Je nach Konfiguration Ihres Geräts werden möglicherweise nicht alle Elemente in dieser Gruppe angezeigt.

MDI

Mit MDI können G-Code-Befehle manuell eingegeben werden. Wenn das Gerät nicht eingeschaltet ist oder wenn ein Programm läuft, sind die MDI-Steuerungen nicht verfügbar.

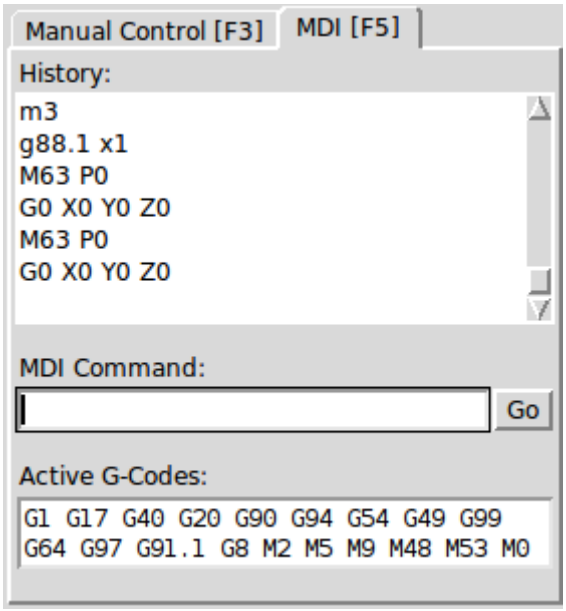


Figure 154. Die MDI-Registerkarte

- *Verlauf* - Hier werden MDI-Befehle angezeigt, die zuvor in dieser Sitzung eingegeben wurden.
- *MDI-Befehl* - Hier können Sie einen G-Code-Befehl eingeben, der ausgeführt werden soll. Führen Sie den Befehl aus, indem Sie Enter drücken oder auf "Go" klicken.
- *Aktive G-Codes* - Dies zeigt die *modalen Codes*, die im Interpreter aktiv sind. Zum Beispiel zeigt *G54* an, dass der *G54-Offset* auf alle eingegebenen Koordinaten angewendet wird. In Auto stellen die aktiven G-Codes die Codes nach dem Vorlesen durch den Interpreter dar.

Vorschub Neufestsetzung (engl. override)

Durch Verschieben dieses Schiebereglers kann der programmierte Vorschub geändert werden. Wenn z.B. ein Programm "F60" verlangt und der Schieberegler auf 120% eingestellt ist, dann ist der resultierende Vorschub 72.

Spindeldrehzahl-Anpassung

Durch Verschieben dieses Schiebereglers kann die programmierte Spindeldrehzahl geändert werden. Wenn ein Programm beispielsweise S8000 anfordert und der Schieberegler auf 80% eingestellt ist, beträgt die resultierende Spindeldrehzahl 6400. Dieser Punkt erscheint nur, wenn der HAL-Pin *spindle.0.speed-out* angeschlossen ist.

Jog-Geschwindigkeit

Durch Bewegen dieses Schiebereglers kann die Geschwindigkeit des Joggens geändert werden. Zum Beispiel, wenn der Schieberegler auf 1 Zoll / min eingestellt ist, dann wird ein 0,01-Zoll-Joggen in etwa 0,6 Sekunden oder 1/100 einer Minute abgeschlossen. In der Nähe der linken Seite (langsames Joggen) sind die Werte eng beieinander angeordnet, während sie in der Nähe der rechten Seite (schnelle Jogs) viel weiter voneinander entfernt sind, was eine breite Palette von Jog-Geschwindigkeiten mit feiner Kontrolle ermöglicht, wenn es am wichtigsten ist.

Auf Maschinen mit Drehachse wird ein zweiter Jog-Speed-Slider angezeigt. Dieser Schieberegler legt die Jog-Rate für die Drehachsen (A, B und C) fest.

Max. Geschwindigkeit

By moving this slider, the maximum velocity can be set. This caps the maximum velocity for all programmed moves except spindle-synchronized moves and moves of angular axes only.

10.1.4. Tastatursteuerung

Fast alle Aktionen in AXIS können über die Tastatur ausgeführt werden. Eine vollständige Liste der Tastaturkürzel finden Sie in der AXIS-Kurzreferenz, die Sie über Hilfe > Kurzreferenz aufrufen können. Viele der Tastenkombinationen sind im MDI-Modus nicht verfügbar.

Vorschub-Neufestsetzung (engl. override)-Tasten

NOTE	Einzelheiten zur spanischen Tastaturbelegung entnehmen Sie bitte der übersetzten Dokumentation.

Die Vorschub-Override-Tasten verhalten sich im manuellen Modus anders. Die Tasten *12345678 wählen eine Achse aus, wenn diese programmiert ist. Wenn Sie 3 Achsen haben, wählt ' die Achse 0, 1 die Achse 1 und 2 die Achse 2. Die übrigen Zifferntasten stellen weiterhin den Vorschub-Override ein. Wenn Sie ein Programm ausführen, stellt '1234567890* den Vorschub-Override auf 0% - 100% ein.

Die am häufigsten verwendeten Tastaturkürzel sind in der folgenden Tabelle aufgeführt:

Table 56. Häufigste Tastaturkürzel

Tastenkombination	Ergriffene Maßnahmen	Modus
F1	Notaus ein-/ausschalten	Jede (engl. any)
F2	Maschine ein-/ausschalten	Jede (engl. any)

Tastenkombination	Ergriffene Maßnahmen	Modus
`, 1 .. 9, 0	Vorschub-Override von 0% bis 100% einstellen	Variiert
X, `	Erste Achse aktivieren	Handbuch
Y, 1	Zweite Achse aktivieren	Handbuch
Z, 2	Dritte Achse aktivieren	Handbuch
A, 3	Vierte Achse aktivieren	Handbuch
I	Jog-Inkrement auswählen	Handbuch
C	Kontinuierliches Joggen	Handbuch
Steuerung-Pos1 (engl. Home)	Referenzfahrt durchführen	Handbuch
Ende	Touch off: G5x Offset für aktive Achse setzen	Handbuch
Links, Rechts	Erste Achse joggen	Handbuch
Hoch, Runter	Zweite Achse joggen	Handbuch
Bild Hoch, Bild Runter (engl. Pg Up, Pg Dn)	Joggen der dritten Achse	Handbuch
[,]	Vierte Achse joggen	Handbuch
O	Datei öffnen	Handbuch
Steuerung-R	Datei neu laden	Handbuch
R	Datei ausführen	Handbuch
P	Ausführung anhalten	Auto
S	Ausführung fortsetzen	Auto
Esc	Ausführung stoppen	Auto
Steuerung-K	Backplot löschen	Auto/Manuell
V	Wechseln zwischen voreingestellten Ansichten	Auto/Manuell
Umschalttaste-Links,Rechts	Eilgang X-Achse	Handbuch
Umschalttaste-Hoch/Runter	Eilgang Y-Achse	Handbuch
Umschalt-Bild auf, Bild ab	Eilgang Z-Achse	Handbuch
@	Umschalten Ist/Befehl	Jede (engl. any)

Tastenkombination	Ergriffene Maßnahmen	Modus
#	Umschalten Relativ/Maschine	Jede (engl. any)

10.1.5. LinuxCNC-Status anzeigen (`linuxcnctop`)

AXIS enthält ein Programm namens *linuxcnctop*, das einige der Details des LinuxCNC-Status anzeigt. Sie können dieses Programm ausführen, indem Sie Maschine > LinuxCNC-Status anzeigen aufrufen

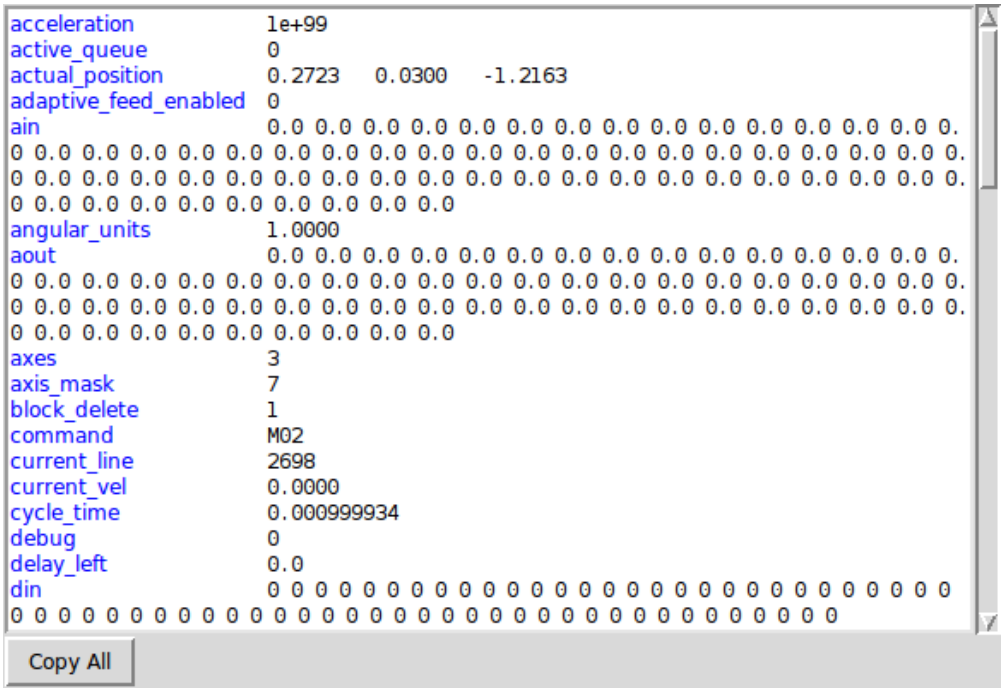


Figure 155. LinuxCNC-Statusfenster

Der Name jedes Elements wird in der linken Spalte angezeigt. Der aktuelle Wert wird in der rechten Spalte angezeigt. Wenn sich der Wert kürzlich geändert hat, wird er rot unterlegt angezeigt.

10.1.6. MDI-Schnittstelle

AXIS enthält ein Programm namens `mdi`, engl. Abkürzung für Manuelle Dateneingabe (manual data input), das die Eingabe von MDI-Befehlen im Textmodus in eine laufende LinuxCNC-Sitzung ermöglicht. Sie können dieses Programm direkt von der UNIX Kommandozeile ausführen, indem Sie ein Terminal öffnen und eingeben:

```
mdi
```

Sobald es läuft, wird die Eingabeaufforderung *MDI*> angezeigt. Bei Eingabe einer leere Zeile wird die aktuelle Position der Maschine angezeigt. Ein eingegebener Befehl wird an LinuxCNC gesendet, um ausgeführt zu werden.

Dies ist eine Beispielsitzung von mdi:

```
$ mdi
```

```
MDI>
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI> G1 F5 X1
MDI>
(0.5928500000000374, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI>
(1.0000000000000639, 0.0, 0.0, 0.0, 0.0, 0.0)
```

10.1.7. axis-remote

AXIS enthält ein Programm namens *axis-remote*, das bestimmte Befehle an einen laufenden AXIS senden kann. Die verfügbaren Befehle werden durch Ausführen von *axis-remote --help* angezeigt und umfassen die Überprüfung, ob AXIS läuft (*--ping*), das Laden einer Datei nach Namen, das erneute Laden der aktuell geladenen Datei (*--reload*) und das Beenden von AXIS (*--quit*).

10.1.8. Manueller Werkzeugwechsel

LinuxCNC enthält eine nicht-Echtzeit-HAL-Komponente namens *hal_manualtoolchange*, die ein Fenster zeigt, das Ihnen sagt, welches Werkzeug erwartet wird, wenn ein *M6*-Befehl ausgegeben wird. Nach dem Drücken der Schaltfläche OK wird die Ausführung des Programms fortgesetzt.

Die Komponente hal_manualtoolchange enthält einen HAL-Pin für eine Taste, die mit einer physischen Taste verbunden werden kann, um den Werkzeugwechsel abzuschließen und die Fensteraufforderung zu entfernen (hal_manualtoolchange.change_button).

Die HAL-Konfigurationsdatei *lib/hallib/axis_manualtoolchange.hal* enthält die HAL-Befehle, die zur Verwendung dieser Komponente erforderlich sind.

hal_manualtoolchange kann auch verwendet werden, wenn AXIS nicht als GUI verwendet wird. Diese Komponente ist besonders nützlich, wenn Sie voreinstellbare Werkzeuge haben und die Werkzeugtabelle verwenden.

NOTE Wichtiger Hinweis: Eilgänge werden nach der Ausgabe eines T<n> bis zum nächsten Vorschub nach dem M6 nicht in der Vorschau angezeigt. Dies kann für die meisten Anwender sehr verwirrend sein. Um diese Funktion für den aktuellen Werkzeugwechsel auszuschalten, programmieren Sie ein G1 ohne Vorschub nach dem T<n>.

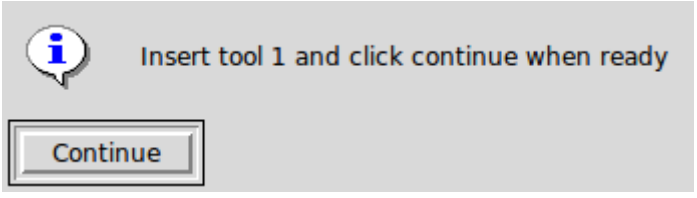


Figure 156. Fenster für manuellen Werkzeugwechsel

10.1.9. Python Module

AXIS enthält mehrere Python-Module, die für andere nützlich sein können. Für weitere Informationen über eines dieser Module verwenden Sie *pydoc <Modulname>* oder lesen Sie den Quellcode. Zu diesen

Modulen gehören:

- emc' ermöglicht den Zugriff auf die LinuxCNC Befehls-, Status- und Fehlerkanäle
- *gcode* bietet Zugriff auf den RS274NGC-Interpreter
- *rs274* bietet zusätzliche Tools für die Arbeit mit RS274NGC-Dateien
- *hal* ermöglicht die Erstellung von in Python geschriebenen nicht-Echtzeit-HAL-Komponenten
- *_togl* stellt ein OpenGL-Widget bereit, das in Tkinter-Anwendungen verwendet werden kann

Um diese Module in Ihren eigenen Skripten zu verwenden, müssen Sie sicherstellen, dass sich das Verzeichnis, in dem sie sich befinden, im Modulpfad von Python befindet. Wenn Sie eine installierte Version von LinuxCNC ausführen, sollte dies automatisch geschehen. Wenn Sie "in-place" laufen, können Sie dies mit "scripts/rip-environment" tun.

10.1.10. Nutzung von AXIS im Drehmaschinenmodus

Durch Einfügen der Zeile *LATHE = 1* in den Abschnitt [DISPLAY] der INI-Datei wählt AXIS den Drehmaschinenmodus. Die Y-Achse wird in den Koordinatenanzeigen nicht angezeigt, die Ansicht wird so geändert, dass die Z-Achse nach rechts und die X-Achse zum unteren Rand des Bildschirms zeigt, und mehrere Steuerelemente (z. B. die für voreingestellte Ansichten) werden entfernt. Die Koordinatenanzeigen für X werden durch Durchmesser und Radius ersetzt.

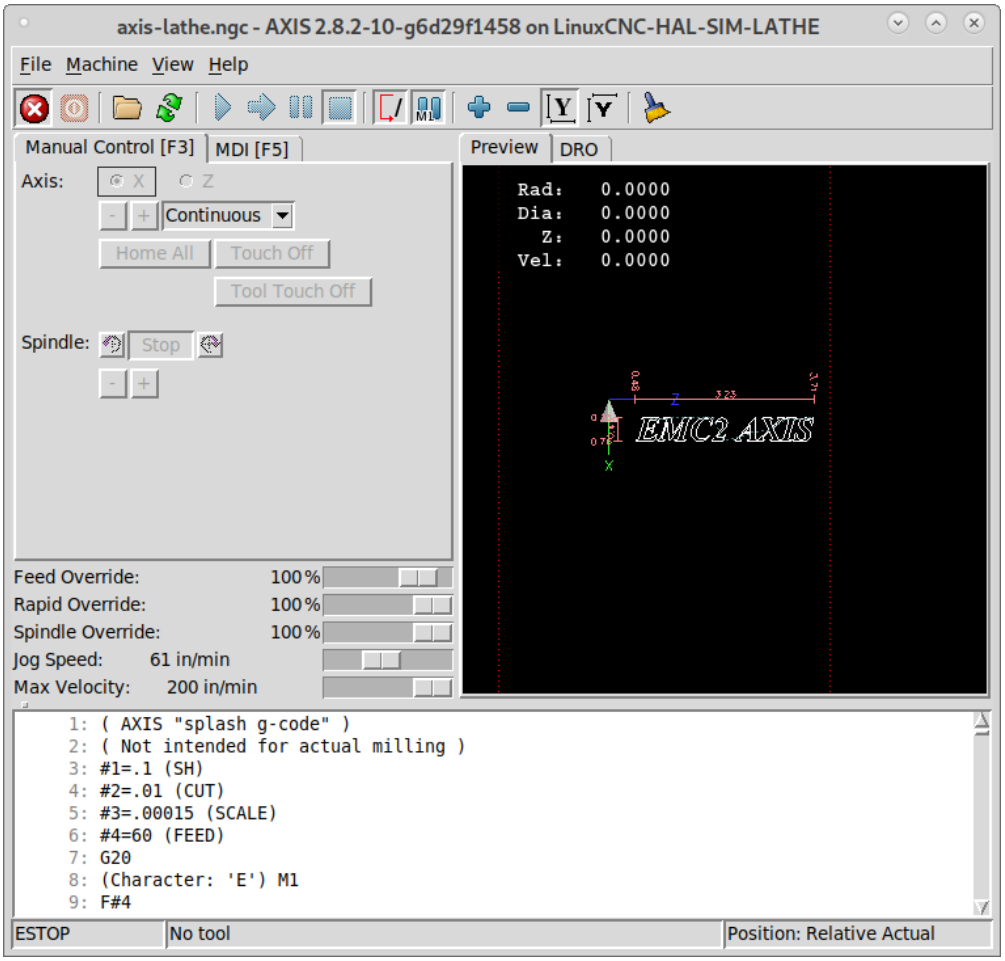


Figure 157. AXIS-Drehmaschinenmodus

Durch Drücken von *V* wird die gesamte Datei angezeigt, sofern eine solche geladen ist.

Im Drehmaschinenmodus wird die Form des geladenen Werkzeugs (falls vorhanden) angezeigt.

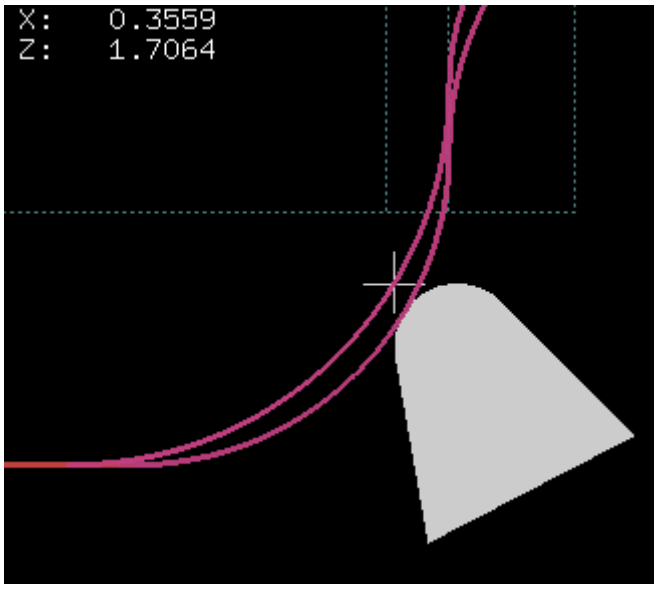


Figure 158. Drehwerkzeug-Form

Um die Anzeige in eine Drehbank mit hinterem Werkzeug zu ändern, müssen Sie sowohl *LATHE = 1* als auch *BACK_TOOL_LATHE = 1* in der Sektion [DISPLAY] eingeben. Dadurch wird die Ansicht umgedreht und das Werkzeug auf die Rückseite der Z-Achse gelegt.

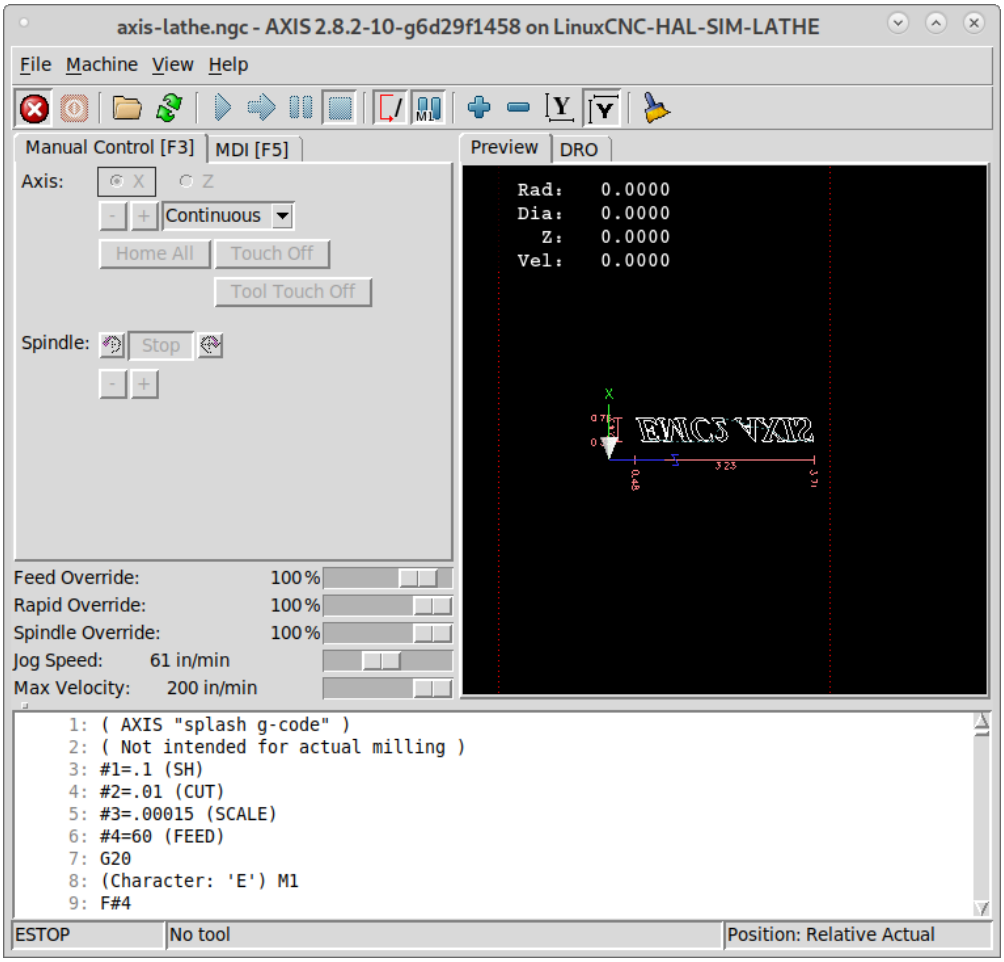


Figure 159. Form des hinteren Drehwerkzeugs

10.1.11. Verwendung von AXIS im Modus Schaumstoffschneiden (engl. foam cutting mode)

Durch Einfügen der Zeile *FOAM = 1* in den [DISPLAY]-Abschnitt der INI-Datei wählt AXIS den Schaumschneidemodus. In der Programmvorschau werden die XY-Bewegungen in einer Ebene und die UV-Bewegungen in einer anderen Ebene angezeigt. In der Live-Darstellung werden Linien zwischen entsprechenden Punkten auf der XY-Ebene und der UV-Ebene gezeichnet. Die speziellen Kommentare (XY_Z_POS) und (UV_Z_POS) legen die Z-Koordinaten dieser Ebenen fest, die standardmäßig 0 und 1,5 Maschineneinheiten betragen.

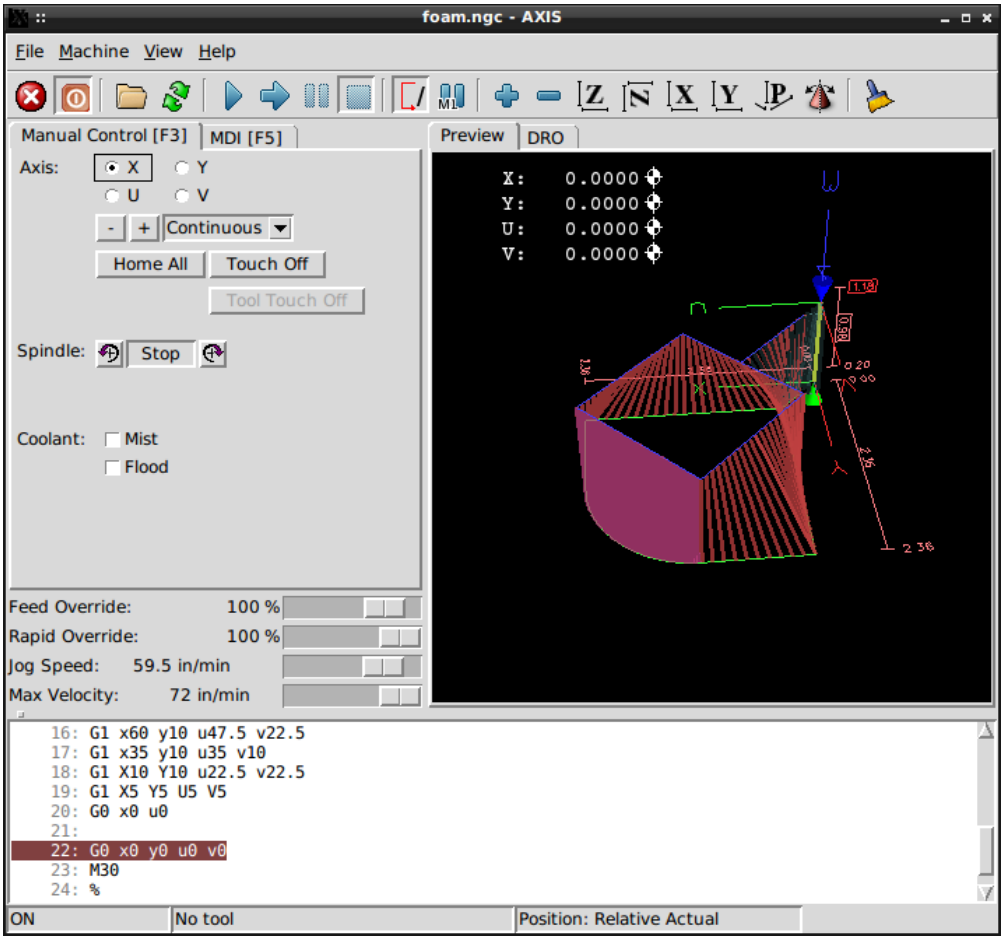


Figure 160. Modus Schaumstoffschneiden

10.1.12. Erweiterte Konfiguration

Wenn AXIS gestartet wird, werden die HAL-Pins für die grafische Benutzeroberfläche erstellt und die in der INI-Datei genannte HAL-Datei ausgeführt: *[HAL]POSTGUI_HALFILE=<Dateiname>*. Typischerweise ist *<Dateiname>* der Basisname der Konfiguration + *_postgui* + *.hal*, z.B. *lathe_postgui.hal*, kann aber jeder beliebige Dateiname sein. Diese Befehle werden nach der Erstellung des Bildschirms ausgeführt und garantieren, dass die HAL-Pins des Widgets verfügbar sind. Sie können mehrere Zeilen mit *POSTGUI_HALFILE=<Dateiname>* in der INI haben. Sie werden nacheinander in der Reihenfolge ausgeführt, in der sie erscheinen.

Weitere Informationen zu den Einstellungen in der INI-Datei der Funktionsweise von AXIS, finden Sie im Kapitel INI-Konfiguration zur Display Section.

Programm-Filter

AXIS hat die Möglichkeit, geladene Dateien durch ein "Filterprogramm" zu schicken. Dieser Filter kann jede gewünschte Aufgabe erfüllen: Etwas so Einfaches wie sicherzustellen, dass die Datei mit "M2" endet, oder etwas so Kompliziertes wie die Erzeugung von G-Code aus einem Bild.

Der Abschnitt [FILTER] der INI-Datei steuert, wie die Filter funktionieren. Schreiben Sie zunächst für jeden Dateityp eine PROGRAM_EXTENSION-Zeile. Dann geben Sie das Programm an, das für jeden Dateityp ausgeführt werden soll. Dieses Programm erhält den Namen der Eingabedatei als erstes Argument und muss rs274ngc-Code in die Standardausgabe schreiben. Diese Ausgabe ist das, was im Textbereich angezeigt wird, in der Vorschau im Anzeigebereich, und dann auch von LinuxCNC ausgeführt wird. Die folgenden Zeilen fügen Unterstützung für den in LinuxCNC enthaltenen "image-to-gcode" (engl. für Bild zu G-Code) -Konverter hinzu:

```
[FILTER]
PROGRAM_EXTENSION = .png,.gif Greyscale Depth Image
png = image-to-gcode
gif = image-to-gcode
```

Es ist auch möglich, einen Interpreter anzugeben:

```
PROGRAM_EXTENSION = .py Python Script
py = python
```

Auf diese Weise kann jedes Python-Skript geöffnet werden, und seine Ausgabe wird als G-Code behandelt. Ein solches Beispielskript ist unter "nc_files/holecircle.py" verfügbar. Dieses Skript erzeugt G-Code für das Bohren einer Reihe von Löchern entlang des Umfangs eines Kreises.

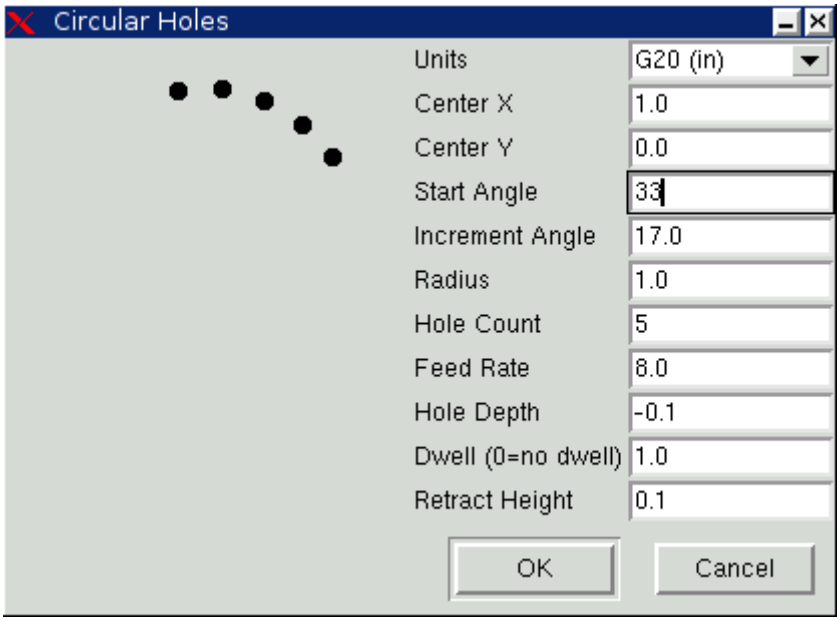


Figure 161. Kreisförmige Löcher

Wenn die Umgebungsvariable AXIS_PROGRESS_BAR gesetzt ist, werden in stderr Zeilen der Form

```
FILTER_PROGRESS=%d
```

setzt den AXIS-Fortschrittsbalken auf den angegebenen Prozentsatz. Diese Funktion sollte von jedem Filter verwendet werden, der lange läuft.

Die X-Ressourcen-Datenbank

Die Farben der meisten Elemente der AXIS-Benutzeroberfläche können über die X-Ressourcen-Datenbank angepasst werden. Die Beispieldatei *axis_light_background* ändert die Farben des Backplot-Fensters in ein Schema "dunkle Linien auf weißem Hintergrund" und dient auch als Referenz für die konfigurierbaren Elemente im Anzeigebereich. Die Beispieldatei *axis_big_dro* ändert die Positionsanzeige in eine größere Schriftart. So verwenden Sie diese Dateien:

```
xrdb -merge /usr/share/doc/emc2/axis_light_background

xrdb -merge /usr/share/doc/emc2/axis_big_dro
```

Informationen zu den anderen Elementen, die in Tk-Anwendungen konfiguriert werden können, finden Sie auf den Manpages von Tk.

Da moderne Desktop-Umgebungen automatisch einige Einstellungen in der X-Ressourcen-Datenbank vornehmen, die sich nachteilig auf AXIS auswirken, werden diese Einstellungen standardmäßig ignoriert. Damit die Elemente der X-Ressourcen-Datenbank die AXIS-Standardeinstellungen außer Kraft setzen, fügen Sie die folgende Zeile in Ihre X-Ressourcen ein:

```
*AXIS*optionLevel: widgetDefault
```

Dies bewirkt, dass die eingebauten Optionen auf der Optionsebene *widgetDefault* erstellt werden, so dass X-Ressourcen (die der Ebene *userDefault* angehören) sie außer Kraft setzen können.

Jogwheel

Um die Interaktion von AXIS mit einem physischen Jogwheel zu verbessern, wird die aktuell aktive Achse, die in der GUI ausgewählt wurde, auch an einen *HAL-Pin* mit einem Namen wie *axisui.jog.x* gemeldet. Außer für eine kurze Zeit nach dem Wechsel der aktuellen Achse ist jeweils nur einer dieser Pins *TRUE*, die anderen bleiben *FALSE*.

Nachdem AXIS diese HAL-Pins erstellt hat, führt es die HAL-Datei aus, die mit: [HAL]POSTGUI_HALFILE deklariert ist. Was unterscheidet sich von [HAL]HALFILE, die nur einmal verwendet werden kann.

~/.axisrc

Wenn sie existiert, wird der Inhalt von `~/.axisrc` als Python-Quellcode ausgeführt, kurz bevor die AXIS-GUI angezeigt wird. Die Details dessen, was in `~/.axisrc` geschrieben werden kann, können sich während des Entwicklungszyklus ändern.

Im Folgenden wird Strg-Q als Tastenkombination für Beenden hinzugefügt.

Beispiel einer .axisrc-Datei

```
root_window.bind("<Control-q>", "destroy .")
help2.append(("Control-Q", "Quit"))
```

Das folgende Beispiel stoppt den Dialog "Wollen Sie wirklich beenden".

```
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

USER_COMMAND_FILE

Eine konfigurationsspezifische Python-Datei kann mit einer INI-Datei-Einstellung *[DISPLAY]USER_COMMAND_FILE=filename.py* angegeben werden. Wie eine `~/.axisrc`-Datei wird diese Datei kurz vor der Anzeige der AXIS-GUI aufgerufen. Diese Datei ist spezifisch für eine INI-Dateikonfiguration und nicht für das Home-Verzeichnis des Benutzers.

user_live_update()

Die AXIS-GUI enthält eine No-Op-Funktion (Platzhalter) namens *user_live_update()*, die am Ende der update()-Funktion der LivePlotter-Klasse ausgeführt wird. Diese Funktion kann in einem `~/.axisrc` -Python-Skript oder einem *[DISPLAY]USER_COMMAND_FILE*-Python-Skript implementiert werden, um benutzerdefinierte, periodische Aktionen auszuführen. Die Details dieser Funktion hängen von der AXIS-GUI-Implementierung ab und können sich im Laufe des Entwicklungszyklus ändern.

user_hal_pins()

Die AXIS-GUI enthält eine No-Op-Funktion (Platzhalter) namens *user_hal_pins()*.
Sie wird unmittelbar nach dem Aufruf der .axisrc-Datei und unmittelbar vor der Initialisierung von GladeVCP-Panels / eingebetteten Registerkarten ausgeführt.
Diese Funktion kann in einem `~/.axisrc` Python-Skript oder einem *[DISPLAY]USER_COMMAND_FILE* Python-Skript implementiert werden, um benutzerdefinierte HAL-Pins zu erstellen, die das Präfix *axisui.* verwenden.
Verwenden Sie *comp* als Referenz für die HAL-Komponenteninstanz.
HAL comp.ready() wird unmittelbar nach der Rückkehr dieser Funktion aufgerufen.

Externer Editor

Die Menüpunkte Datei > Bearbeiten... und Datei > Werkzeugtabelle bearbeiten... werden nach der Definition des Editors im INI-Abschnitt [DISPLAY] verfügbar. Nützliche Werte sind `EDITOR=gedit` und `EDITOR=gnome-terminal -e vim`. Weitere Informationen finden Sie unter Display Section im Kapitel INI-Konfiguration.

Virtuelle Schalttafel

AXIS can display a custom virtual control panel in either the right side column or the bottom row. Additionally one or more panels may be displayed as embedded tabs. You can program buttons, indicators, data displays and more. For more information, see the PyVCP, the GladeVCP chapters and

QtVCP panels.

Vorschau Kontrolle

Spezielle Kommentare können in die G-Code-Datei eingefügt werden, um zu steuern, wie sich die Vorschau von AXIS verhält. Wenn Sie das Zeichnen der Vorschau einschränken wollen, verwenden Sie diese speziellen Kommentare. Alles, was zwischen (AXIS,hide) und (AXIS,show) liegt, wird während der Vorschau nicht gezeichnet. (AXIS,hide) und (AXIS,show) müssen paarweise verwendet werden, wobei (AXIS,hide) an erster Stelle steht. Alles, was nach einem (AXIS,stop) kommt, wird während der Vorschau nicht gezeichnet.

Diese Kommentare sind nützlich, um die Anzeige der Vorschau zu entschlacken (z. B. kann man bei der Fehlersuche in einer größeren G-Code-Datei die Vorschau für bestimmte Teile, die bereits gut funktionieren, deaktivieren).

- (AXIS,hide) Stoppt die Vorschau (muss zuerst sein)
- (AXIS,show) Setzt die Vorschau fort (muss auf ein *hide* folgen)
- (AXIS,stop) Stoppt die Vorschau von hier bis zum Ende der Datei.
- (AXIS,notify,the_text) Zeigt the_text als Infoanzeige an

Diese Anzeige kann in der AXIS-Vorschau nützlich sein, wenn (Debug-, Nachrichten-) Kommentare nicht angezeigt werden.

The preview can also draw the stock the program is cut from; see the Workpiece section. *View* > *Solid workpiece* chooses how opaque its faces are drawn, from *Off* (the outline alone) to *Full*.

Touch Off mit der Actual Position (tatsächliche Position)

Die Funktion zum Nullpunkt-Setzen (Touch Off) kann optional den Ist-Positionswert der Achse in die Berechnung des Offsets einbeziehen. Dies wird vor allem in Fällen genutzt, in denen eine nicht motorisierte Achse, wie z. B. die Pinole einer Fräsmaschine, über einen Encoder Rückmeldung an LinuxCNC liefert, aber kein Motor zur Bewegungskontrolle vorhanden ist. So kann die Achse (AXIS) für eine solche Achse eine DRO-Anzeige bereitstellen, die eine funktionierende Funktion zum Nullpunkt-Setzen (Touch Off) bietet.

Diese Funktion wird für eine Achse aktiviert, indem der entsprechende Abschnitt *[AXIS_x]* in der .INI-Datei geändert wird. Fügen Sie eine Option mit dem Namen *TOUCHOFF_ACTUAL* hinzu und setzen Sie den Wert auf *PLUS* oder *MINUS*, je nachdem, wie der Ist-Positionswert auf das Offset angewendet werden soll.

Beispiel:

```
[AXIS_Z]
TOUCHOFF_ACTUAL = MINUS
```

Normalerweise wird nur die Soll-Position einer Achse verwendet, um dieses Offset zu setzen, was dazu führt, dass die Funktion nicht korrekt arbeitet, da nicht motorisierte Achsen nie bewegt werden und ihre

Soll-Position daher immer 0 ist.

Die Funktion zum Nullpunkt-Setzen (Touch Off) sendet einen *G10 L20*-Befehl an das MDI, um den neuen Offset-Wert zu setzen. Normalerweise wird dabei einfach der im Dialogfeld eingegebene Wert verwendet. Wenn diese Funktion aktiviert ist, wird je nach Konfiguration der aktuelle Positionswert zum im Dialogfeld eingegebenen Wert addiert oder davon subtrahiert.

10.1.13. Axisui

Um die Interaktion von AXIS mit physischen Jogwheels zu verbessern, wird die aktuell in der GUI ausgewählte Achse auch auf einem Pin mit einem Namen wie *axisui.jog.x* gemeldet. Einer dieser Pins ist immer *TRUE*, die anderen sind *FALSE*. Diese sind dazu gedacht, die Jog-Aktivierungspins von Motion zu steuern.

Axisui-Pins

AXIS verfügt über HAL-Pins, die anzeigen, welcher Jog-Radio-Button auf der Registerkarte "Manuelle Steuerung" ausgewählt ist.

```
Type  Dir  Name
bool  OUT  axisui.jog.x
bool  OUT  axisui.jog.y
bool  OUT  axisui.jog.z
bool  OUT  axisui.jog.a
bool  OUT  axisui.jog.b
bool  OUT  axisui.jog.c
bool  OUT  axisui.jog.u
bool  OUT  axisui.jog.v
bool  OUT  axisui.jog.w
```

AXIS verfügt über einen HAL-Pin zur Anzeige der auf der Registerkarte "Manuell" ausgewählten Schrittweite.

```
Type  Dir Name
real  OUT axisui.jog.increment
```

AXIS hat einen HAL-Ausgangspin, der anzeigt, wenn ein Abbruch stattgefunden hat. Der Pin *axisui.abort* wird *TRUE* und kehrt nach 0,3 ms auf *FALSE* zurück.

```
Type  Dir    Name
bool  OUT    axisui.abort
```

AXIS verfügt über einen HAL-Ausgabe-Pin, der anzeigt, wenn ein Fehler aufgetreten ist. Der Pin *axisui.error* bleibt *TRUE*, bis alle Fehlerbenachrichtigungen geschlossen wurden.

```
Type  Dir    Name
bool  OUT    axisui.error
```

AXIS verfügt über HAL-Eingangspins, um die Popup-Benachrichtigungen nach Fehlern und Informationen zu löschen.

```
Type  Dir    Name
bool  IN     axisui.notifications-clear
bool  IN     axisui.notifications-clear-error
bool  IN     axisui.notifications-clear-info
```

AXIS verfügt über einen HAL-Eingangspin, der die Funktion "Pause/Resume" deaktiviert/aktiviert.

```
Type  Dir    Name
bool  IN     axisui.resume-inhibit
```

10.1.14. Hinweise zur AXIS-Anpassung

AXIS ist eine ziemlich große und schwer zu durchdringende Codebasis. Das ist hilfreich, um den Code stabil zu halten, macht es aber schwierig, ihn anzupassen.
Hier werden wir Codeschnipsel zeigen, um das Verhalten oder die Darstellung des Bildschirms zu ändern. Bitte beachten Sie, dass sich der interne Code von AXIS von Zeit zu Zeit ändern kann.
Es ist nicht garantiert, dass diese Schnipsel weiterhin funktionieren - sie müssen möglicherweise angepasst werden.

Die Update-Funktion

In AXIS gibt es eine Funktion namens user_live_update, die jedes Mal aufgerufen wird, wenn AXIS sich selbst aktualisiert. Sie können diese Funktion verwenden, um Ihre eigenen Funktionen zu aktualisieren.

```
# continuous update function
def user_live_update():
    print('i am printed every update...')
```

Deaktivieren des Schließen-Dialogs

```
# Deaktivieren Sie den "Do you want to close"-Dialog
root_window.tk.call("wm", "protocol",".", "WM_DELETE_WINDOW", "destroy .")
```

Ändern Sie die Textschriftart

```
# Schriftart ändern

font = 'sans 11'
fname,fsize = font.split()
root_window.tk.call('font','configure','TkDefaultFont','-family',fname,'-size',fsize)

# den Text in den Tabs so umgestalten, dass er die Größe der neuen Standardschriftart
annimmt

root_window.tk.call('.pane.top.tabs','itemconfigure','manual','-text',' Manual - F3 ')
root_window.tk.call('.pane.top.tabs','itemconfigure','mdi','-text',' MDI - F5 ')
root_window.tk.call('.pane.top.right','itemconfigure','preview','-text',' Preview ')
root_window.tk.call('.pane.top.right','itemconfigure','numbers','-text',' DRO ')
```

```
# G-Code-Schriftart ist unabhängig

root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue')
#root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue','-font',font)
#root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue','-
font',font,'-height','12')
```

Ändern der Rapid Rate mit Tastenkombinationen

```
# Verwenden Sie Control + ` oder 1-0 als Tastaturkürzel für die rapid rate und behalten
Sie ` oder 1-0 für feedrate
# fügt auch Text zur Kurzreferenz in der Hilfe hinzu

help1.insert(10,("Strg+ `,1..9,0", _("Set Rapid Override from 0% to 100%")),)

root_window.bind('<Control-Key-quoteleft>',lambda event: set_rapidrate(0))
root_window.bind('<Control-Key-1>',lambda event: set_rapidrate(10))
root_window.bind('<Control-Key-2>',lambda event: set_rapidrate(20))
root_window.bind('<Control-Key-3>',lambda event: set_rapidrate(30))
root_window.bind('<Control-Key-4>',lambda event: set_rapidrate(40))
root_window.bind('<Control-Key-5>',lambda event: set_rapidrate(50))
root_window.bind('<Control-Key-6>',lambda event: set_rapidrate(60))
root_window.bind('<Control-Key-7>',lambda event: set_rapidrate(70))
root_window.bind('<Control-Key-8>',lambda event: set_rapidrate(80))
root_window.bind('<Control-Key-9>',lambda event: set_rapidrate(90))
root_window.bind('<Control-Key-0>',lambda event: set_rapidrate(100))
root_window.bind('<Key-quoteleft>',lambda event: set_feedrate(0))
root_window.bind('<Key-1>',lambda event: set_feedrate(10))
root_window.bind('<Key-2>',lambda event: set_feedrate(20))
root_window.bind('<Key-3>',lambda event: set_feedrate(30))
root_window.bind('<Key-4>',lambda event: set_feedrate(40))
root_window.bind('<Key-5>',lambda event: set_feedrate(50))
root_window.bind('<Key-6>',lambda event: set_feedrate(60))
root_window.bind('<Key-7>',lambda event: set_feedrate(70))
root_window.bind('<Key-8>',lambda event: set_feedrate(80))
root_window.bind('<Key-9>',lambda event: set_feedrate(90))
root_window.bind('<Key-0>',lambda event: set_feedrate(100))
```

Lesen der INI-Datei

```
# INI-Dateielement lesen
machine = inifile.find('EMC','MACHINE')
print('machine name =',machine)
```

Lesen des LinuxCNC-Status

```
# LinuxCNC-Status kann aus s gelesen werden.
print(s.actual_position)
print(s.paused)
```

Ändern der aktuellen Ansicht

```
# Legen Sie die Ansicht der Vorschau fest.
# gültige Ansichten sind view_x view_y view_y2 view_z view_z2 view_p
commands.set_view_z()
```

Erstellen neuer AXISUI HAL-Pins

```
def user_hal_pins():
    comp.newpin('my-new-in-pin', hal.Type.BOOL, hal.Dir.IN)
    comp.ready()
```

Neue HAL-Komponente und Pins erstellen

```
# Komponente erstellen

mycomp = hal.component('my_component')
mycomp.newpin('idle-led',  hal.Type.BOOL, hal.Dir.IN)
mycomp.newpin('pause-led', hal.Type.BOOL, hal.Dir.IN)
mycomp.ready()

# Pins verbinden

hal.new_sig('idle-led', hal.Type.BOOL)
hal.connect('halui.program.is-idle','idle-led')
hal.connect('my_component.idle-led','idle-led')

# Pin setzen

hal.set_p('meine_Komponente.pause-led','1')

# Pin auslesen (engl. get) ab Version 2.8

value = hal.get_value('halui.program.is-idle')
print('value is a',type(value),'value of',value)
```

Tabs wechseln mit HAL-Pins

```
# HAL Pins von einem GladeVCP-Panel werden nicht bereit sein, wenn user_live_update
ausgeführt wird.
# um sie zu lesen, müssen Sie sie in einen try/except-Block setzen

# Das folgende Beispiel geht von 5 HAL-Tasten in einem GladeVCP-Panel aus, die zum
Umschalten
# die Registerkarten von AXIS.
# Die Namen der Schaltflächen sind 'manual-tab', 'mdi-tab', 'preview-tab', 'dro-tab',
'user0-tab'.
# Die Registerkarte "user_0" wäre, falls vorhanden, die erste in GladeVCP eingebettete
Registerkarte.

# LinuxCNC ab Version 2.8
```

```
def user_live_update():
    try:
        if hal.get_value('gladevcp.manual-tab'):
            root_window.tk.call('.pane.top.tabs','raise','manual')
        elif hal.get_value('gladevcp.mdi-tab'):
            root_window.tk.call('.pane.top.tabs','raise','mdi')
        elif hal.get_value('gladevcp.preview-tab'):
            root_window.tk.call('.pane.top.right','raise','preview')
        elif hal.get_value('gladevcp.numbers-tab'):
            root_window.tk.call('.pane.top.right','raise','numbers')
        elif hal.get_value('gladevcp.user0-tab'):
            root_window.tk.call('.pane.top.right','raise','user_0')
    except:
        pass
```

Hinzufügen einer GOTO Referenzpunkt (engl. Home)-Taste

```
def goto_home(axis):
    if s.interp_state == linuxcnc.INTERP_IDLE:
        home = inifile.find('JOINT_' + str(inifile.find('TRAJ', 'COORDINATES').upper
().index(axis)), 'HOME')
        mode = s.task_mode
        if s.task_mode != linuxcnc.MODE_MDI:
            c.mode(linuxcnc.MODE_MDI)
        c.mdi('G53 G0 ' + axis + home)

# einen Button für die Y-Achse erzeugen
root_window.tk.call('button','.pane.top.tabs.fmanual.homey','-text','Home Y','-command',
'goto_home Y','-height','2')

# Platzieren des Button
root_window.tk.call('grid','.pane.top.tabs.fmanual.homey','-column','1','-row','7','
-columnspan','2','-padx','4','-sticky','w')

# jede Funktion, die aus Tcl aufgerufen wird, muss zu TclCommands hinzugefügt werden
TclCommands.goto_home = goto_home
Befehle = TclCommands(root_window)
```

Button zum manuellen Rahmen hinzufügen

```
# Erstellen eines neuen Button und einfügen in den manuellen Rahmen

root_window.tk.call('button','.pane.top.tabs.fmanual.mybutton','-text','My Button','
-command','mybutton_clicked','-height','2')
root_window.tk.call('grid','.pane.top.tabs.fmanual.mybutton','-column','1','-row','6','
-columnspan','2','-padx','4','-sticky','w')

# Die obigen senden den Befehl "mybutton_clicked", wenn sie angeklickt werden
# Weitere Optionen sind das Binden eines Druck- oder Freigabebefehls (oder beides) an die
Schaltfläche
# diese können zusätzlich oder anstelle des angeklickten Befehls sein,
# dann '-command','mybutton_clicked' aus der ersten Zeile löschen.
```

```
# Button-1 = linke Maustaste, 2 = rechte oder 3 = mittlere Maustaste

root_window.tk.call('bind','.pane.top.tabs.fmanual.mybutton','<Button-1>',
'mybutton_pressed')
root_window.tk.call('bind','.pane.top.tabs.fmanual.mybutton','<ButtonRelease-1>',
'mybutton_released')

# Funktionen, die von den Buttons aufgerufen werden

def mybutton_clicked():
    print('mybutton was clicked')
def mybutton_pressed():
    print('mybutton was pressed')
def mybutton_released():
    print('mybutton was released')

# jede Funktion, die von Tcl aufgerufen wird, muss zu TclCommands hinzugefügt werden

TclCommands.mybutton_clicked = mybutton_clicked
TclCommands.mybutton_pressed = mybutton_pressed
TclCommands.mybutton_released = mybutton_released
commands = TclCommands(root_window)
```

Interne Variablen lesen

```
# die folgenden Variablen können aus der vars-Instanz gelesen werden

print(vars.machine.get())
print(vars.emcini.get())

    active_codes            = StringVar
    block_delete            = BooleanVar
    brake                   = BooleanVar
    coord_type              = IntVar
    display_type            = IntVar
    dro_large_font          = IntVar
    emcini                  = StringVar
    exec_state              = IntVar
    feedrate                = IntVar
    flood                   = BooleanVar
    grid_size               = DoubleVar
    has_editor              = IntVar
    has_ladder              = IntVar
    highlight_line          = IntVar
    interp_pause            = IntVar
    interp_state            = IntVar
    ja_rbutton              = StringVar
    jog_aspeed              = DoubleVar
    jog_speed               = DoubleVar
    kinematics_type         = IntVar
    linuxcnctop_command     = StringVar
    machine                 = StringVar
    max_aspeed              = DoubleVar
    max_maxvel              = DoubleVar
    max_queued_mdi_commands = IntVar
```

```
    max_speed               = DoubleVar
    maxvel_speed            = DoubleVar
    mdi_command             = StringVar
    metric                  = IntVar
    mist                    = BooleanVar
    motion_mode             = IntVar
    on_any_limit            = BooleanVar
    optional_stop           = BooleanVar
    override_limits         = BooleanVar
    program_alpha           = IntVar
    queued_mdi_commands     = IntVar
    rapidrate               = IntVar
    rotate_mode             = BooleanVar
    running_line            = IntVar
    show_distance_to_go     = IntVar
    show_extents            = IntVar
    show_live_plot          = IntVar
    show_machine_limits     = IntVar
    show_machine_speed      = IntVar
    show_program            = IntVar
    show_pyvcppanel         = IntVar
    show_rapids             = IntVar
    show_tool               = IntVar
    show_offsets            = IntVar
    spindledir              = IntVar
    spindlerate             = IntVar
    task_mode               = IntVar
    task_paused             = IntVar
    task_state              = IntVar
    taskfile                = StringVar
    teleop_mode             = IntVar
    workpiece_opacity       = DoubleVar
    tool                    = StringVar
    touch_off_system        = StringVar
    trajcoordinates         = StringVar
    tto_g11                 = BooleanVar
    view_type               = IntVar
```

Widgets ausblenden

```
# ein Widget ausblenden
# 'grid' oder 'pack' verwenden, je nachdem, wie es ursprünglich platziert wurde
root_window.tk.call('grid','forget','.pane.top.tabs.fmanual.jogf.zerohome.tooltouch')
```

Ändern eines Labels

```
# Label eines Widgets ändern
root_window.tk.call('setup_widget_accel','.pane.top.tabs.fmanual.mist','Downdraft')

# sicherstellen, dass es erscheint (in diesem Fall nur erforderlich, wenn die
Schaltfläche mist ausgeblendet war)
root_window.tk.call('grid','.pane.top.tabs.fmanual.mist','-column','1','-row','5','
-columnspan','2','-padx','4','-sticky','w')
```

Einen bestehenden Befehl umleiten

```
# einen bestehenden Befehl abgreifen
# ursprünglich ruft die Schaltfläche mist die Funktion mist auf
root_window.tk.call('.pane.top.tabs.fmanual.mist','configure','-command',
'hijacked_command')

# Die neue Funktion
def hijacked_command():
    print('abgegriffener mist command')

# Hinzufügen der Funktion zu TclCommands
TclCommands.hijacked_command = hijacked_command
Befehle = TclCommands(root_window)
```

Ändern Sie die DRO-Farbe

```
# dro-Bildschirm ändern
root_window.tk.call('.pane.top.right.fnumbers.text','configure','-foreground','green','
-background','black')
```

Ändern der Buttons der Werkzeugleiste

```
# ändern der Werkzeugleisten-Buttons

buW = '3'
buH = '2'
boW = '3'

root_window.tk.call('.toolbar.machine_estop','configure','-image','','-text','ESTOP','
-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.machine_power','configure','-image','','-text','POWER','
-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.file_open','configure','-image','','-text','OPEN','-width'
,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.reload','configure','-image','','-text','RELOAD','-width'
,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_run','configure','-image','','-text','RUN','-width
',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_step','configure','-image','','-text','STEP','
-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_pause','configure','-image','','-text','PAUSE','
-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_stop','configure','-image','','-text','STOP','
-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_blockdelete','configure','-image','','-text','Skip
/','-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_optpause','configure','-image','','-text','M1','
-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_zoomin','configure','-image','','-text','Zoom+','
-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_zoomout','configure','-image','','-text','Zoom-','
-width',buW,'-height',buH,'-borderwidth',boW)
```

```
root_window.tk.call('.toolbar.view_z','configure','-image','','-text','Top X','-width'
,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_z2','configure','-image','','-text','Top Y','-width'
,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_x','configure','-image','','-text','Right','-width'
,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_y','configure','-image','','-text','Front','-width'
,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_p','configure','-image','','-text','3D','-width',buW,'
-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.rotate','configure','-image','','-text','Rotate','-width'
,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.clear_plot','configure','-image','','-text','Clear','-
width',buW,'-height',buH,'-borderwidth',boW)
```

Plotterfarben ändern

Im RGBA-Format, in dieser Reihenfolge: Joggen, Eilgang, Vorschub, Lichtbogen, Werkzeugwechsel, Messtaster

```
# Plotterfarben ändern
try:
    live_plotter.logger.set_colors((255,0,0,255),
                                   (0,255,0,255),
                                   (0,0,255,255),
                                   (255,255,0,255),
                                   (255,255,255,255),
                                   (0,255,255,255))
except Exception as e:
    print(e)
```

10.2. GMOCCAPY

10.2.1. Einführung

GMOCCAPY ist eine grafische Benutzeroberfläche für LinuxCNC, die für die Verwendung mit einem Touchscreen entwickelt wurde, aber auch auf normalen Bildschirmen mit einer Maus oder Hardware-Tasten und MPG-Rädern verwendet werden kann, da sie HAL-Pins für die häufigsten Anforderungen bereitstellt. Weitere Informationen finden Sie im Folgenden.

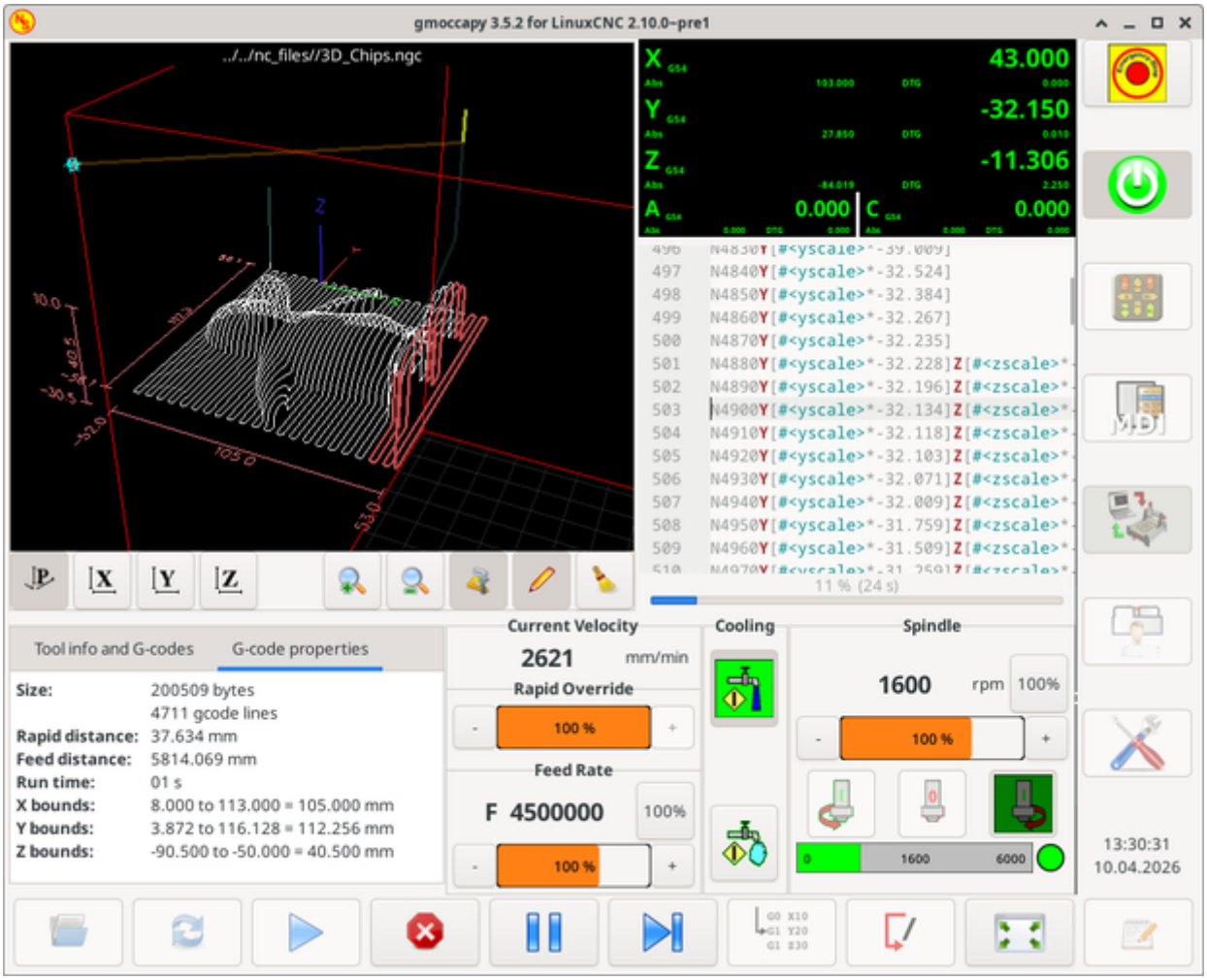
Es bietet die Möglichkeit, bis zu 9 Achsen darzustellen, unterstützt einen Drehmodus für normale und rückseitige Werkzeugdrehungen und kann an nahezu jeden Bedarf angepasst werden, da GMOCCAPY eingebettete Tabs und Seitenpanels unterstützt. Als gutes Beispiel dafür siehe gmoccapy_plasma.

GMOCCAPY 3 unterstützt bis zu 9 Achsen und 9 Gelenke. Da GMOCCAPY 3 im Code geändert wurde, um die Gelenk- / Achsenänderungen in LinuxCNC zu unterstützen, funktioniert es nicht mit LinuxCNC 2.7 oder 2.6!

Es unterstützt eine integrierte virtuelle Tastatur (Onboard- oder Matchbox-Tastatur), so dass keine Hardware-Tastatur oder -Maus benötigt wird, kann aber auch mit dieser Hardware verwendet werden.

GMOCCAPY bietet eine separate Einstellungsseite, um die meisten Einstellungen der GUI zu konfigurieren, ohne Dateien zu bearbeiten.

GMOCCAPY kann sehr einfach lokalisiert werden, da die entsprechenden Dateien von den linuxcnc.po-Dateien getrennt sind, so dass es nicht nötig ist, unnötiges Zeug zu übersetzen. Wenn Sie eine Übersetzung beisteuern möchten, verwenden Sie bitte den webbasierter Übersetzungseditor Weblate. Weitere Informationen finden Sie im Abschnitt Translations



10.2.2. Anforderungen

GMOCCAPY 3 wurde auf Debian Jessie, Debian Stretch und MINT 18 mit LinuxCNC Master und 2.8 Release getestet. Es unterstützt vollständig Gelenk / Achse Änderungen von LinuxCNC, so dass es geeignet als GUI für Scara, Roboter oder jede andere Konfiguration mit mehr Gelenke als Achse. Wenn Sie andere Versionen verwenden, informieren Sie bitte über Probleme und / oder Lösungen auf der LinuxCNC Forum oder die Deutsche CNC Ecke Forum oder LinuxCNC Benutzer Mailingliste.

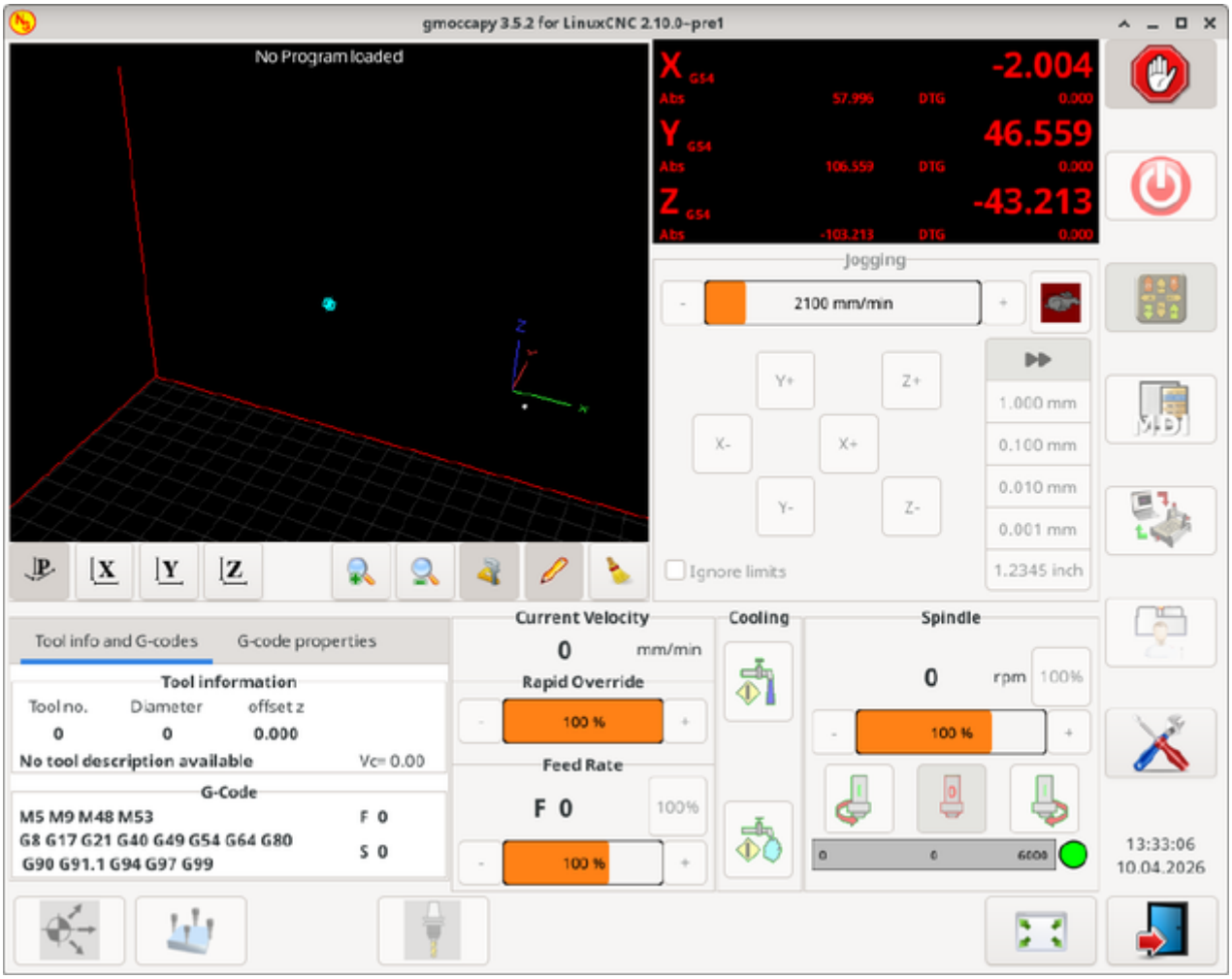
Die minimale Bildschirmauflösung für GMOCCAPY ist **979 x 750 Pixel**, also sollte es auf jeden Standardbildschirm passen. Es wird empfohlen, Bildschirme mit einer Mindestauflösung von 1024x748 zu verwenden. Es gibt auch eine Version für eine Auflösung von 800x600 (eingeführt mit GMOCCAPY 3.4.8).

10.2.3. So erhalten Sie GMOCCAPY

GMOCCAPY 3 ist in der Standarddistribution von LinuxCNC seit Version 2.7 enthalten. Der einfachste Weg, GMOCCAPY auf Ihren Steuerungs-PC zu bekommen, ist also, die ISO herunterzuladen und von der CD/DVD/USB-Stick zu installieren. Dies ermöglicht Ihnen, Aktualisierungen mit den regulären Debian-Paketen zu erhalten.

In den **release notes** können Sie die neuesten Bugfixes und Features verfolgen.

Sie erhalten einen ähnlichen Bildschirm wie den folgenden (das Design kann je nach Ihrer Konfiguration variieren):



10.2.4. Grundeinstellungen

GMOCCAPY 3 unterstützt die folgenden Befehlszeilenoptionen:

- *-user_mode*: Wenn diese Einstellung eingestellt ist, wird der Setup-Button deaktiviert, so dass normale Maschinenbediener die Einstellungen der Maschine nicht bearbeiten können.
- *-logo <Pfad zur Logodatei>*: Wenn das Logo angegeben wird, dann wird die Registerkarte für die Tipptasten im manuellen Modus ausgeblendet. Dies ist nur für Maschinen mit Hardwaretasten für das Jogging und die Schrittweitenauswahl nützlich.

Es gibt eigentlich nicht viel zu konfigurieren, um GMOCCAPY auszuführen, aber es gibt einige Punkte,

die Sie beachten sollten, wenn Sie alle Funktionen der GUI nutzen wollen.

Sie werden eine Reihe von Simulationskonfigurationen (INI-Dateien) finden, die nur die Grundlagen zeigen:

- gmoccapy.ini
- gmoccapy_4_axis.ini
- lathe_configs/gmoccapy_lathe.ini
- lathe_configs/gmoccapy_lathe_imperial.ini
- gmoccapy_left_panel.ini
- gmoccapy_right_panel.ini
- gmoccapy_messages.ini
- gmoccapy_pendant.ini
- gmoccapy_sim_hardware_button.ini
- gmoccapy_tool_sensor.ini
- gmoccapy_with_user_tabs.ini
- gmoccapy_XYZAB.ini
- gmoccapy_XYZAC.ini
- gmoccapy_XYZCW.ini
- gmoccapy-JA/Gantry/gantry_mm.ini
- gmoccapy-JA/scara/scara.ini
- gmoccapy-JA/table-rotary-tilting/xyzac-trt.ini
- und vieles mehr ...

Die Namen sollten den Hauptzweck der verschiedenen INI-Dateien erklären.

Wenn Sie eine bestehende Konfiguration Ihres Rechners verwenden, bearbeiten Sie einfach Ihre INI-Datei entsprechend diesem Dokument.

Schauen wir uns also die INI-Datei genauer an und was Sie einfügen müssen, um GMOCCAPY auf Ihrem Rechner zu verwenden:

Der DISPLAY-Abschnitt

```
[DISPLAY]
DISPLAY = gmoccapy
PREFERENCE_FILE_PATH = gmoccapy_preferences
MAX_FEED_OVERRIDE = 1.5
MAX_SPINDLE_OVERRIDE = 1.2
MIN_SPINDLE_OVERRIDE = 0.5
DEFAULT_SPINDLE_SPEED = 500
LATHE = 1
BACK_TOOL_LATHE = 1
```

```
PROGRAM_PREFIX = ../../nc_files/
```

- *DISPLAY = gmoccapy* - Damit wird LinuxCNC angewiesen, GMOCCAPY zu verwenden.
- PREFERENCE_FILE_PATH - Gibt den Ort und den Namen der zu verwendenden Einstellungsdatei an. In den meisten Fällen wird diese Zeile nicht benötigt, sie wird von GMOCCAPY verwendet, um Ihre Einstellungen der Benutzeroberfläche zu speichern, wie Themen, DRO-Einheiten, Farben und Tastatureinstellungen, usw., siehe Seite für Einstellungen für weitere Details.

NOTE Wenn kein Pfad oder keine Datei angegeben wird, verwendet GMOCCAPY standardmäßig <Ihr_Maschinenname>.pref. Ist kein Maschinenname in Ihrer INI-Datei angegeben, so verwendet es gmoccapy.pref. Die Datei wird in Ihrem Konfigurationsverzeichnis gespeichert, so dass die Einstellungen nicht vermischt werden, wenn Sie mehrere Konfigurationen verwenden. Wenn Sie nur eine Datei für mehrere Rechner verwenden wollen, müssen Sie `PREFERENCE_FILE_PATH` in Ihre INI-Datei aufnehmen.

- *MAX_FEED_OVERRIDE = 1.5* - Legt die maximale Überschreitung/Neufestsetzung (engl. override) des Vorschubs fest. Im angegebenen Beispiel können Sie den Vorschub um 150 % überschreiten.

NOTE Wenn kein Wert angegeben wird, so wird er auf 1,0 gesetzt.

- *MIN_SPINDLE_OVERRIDE = 0.5* and *MAX_SPINDLE_OVERRIDE = 1.2* - Ermöglicht es Ihnen, die Spindelübersteuerung innerhalb einer Grenze von 50% bis 120% zu ändern.

NOTE Wenn keine Werte angegeben sind, wird MIN auf 0,1 und MAX auf 1,0 gesetzt.

- *LATHE = 1* - Legt das Bildschirmlayout für die Steuerung einer Drehmaschine fest.
- *BACK_TOOL_LATHE = 1* - Ist optional und schaltet die X-Achse so um, wie Sie es für eine Drehmaschine mit hinterem Werkzeug benötigen. Auch die Tastenkombinationen werden anders reagieren. Es ist mit GMOCCAPY erlaubt, eine Drehmaschine auch mit zusätzlichen Achsen zu konfigurieren, so dass Sie auch eine XZCW-Konfiguration für eine Drehmaschine verwenden können.

TIP Siehe auch den Abschnitt speziell zu Drehmaschinen.

- *PROGRAM_PREFIX = ../../nc_files/* - Ist der Eintrag, der LinuxCNC/GMOCCAPY mitteilt, wo die NGC-Dateien zu suchen sind.

NOTE Wenn nicht angegeben, sucht Gmoccapy in der folgenden Reihenfolge nach NGC-Dateien: Zunächst `linuxcnc/nc_files` und dann das Heimatverzeichnis (engl. home directory) des Benutzers.

- *DEFAULT_SPINDLE_SPEED* – Startwert für "Starting RPM", falls der Wert nicht in der Einstellungsdatei vorhanden ist oder die Datei fehlt. Hat keine Wirkung, wenn eine gültige Einstellungsdatei vorhanden ist.
- *MIN_ANGULAR_VELOCITY* - Legt die minimale Jog-Geschwindigkeit der Maschine für rotierende

Achsen fest.

- *MAX_ANGULAR_VELOCITY* - Legt die maximale Jog-Geschwindigkeit der Maschine für rotierende Achsen fest.
- *DEFAULT_ANGULAR_VELOCITY* - Legt die voreingestellte Jog-Geschwindigkeit der Maschine für rotierende Achsen fest.

Der TRAJ-Abschnitt

- *DEFAULT_LINEAR_VELOCITY = 85.0* - Voreinstellung der Jog-Geschwindigkeit der Maschine.

> **NOTE** Ist dieser Wert nicht angegeben, wird die Hälfte von *MAX_LINEAR_VELOCITY* verwendet. Wird auch dieser Wert nicht angegeben, so wird er auf 180 gesetzt.

- *MAX_LINEAR_VELOCITY = 230.0* - Legt die maximale Geschwindigkeit der Maschine fest. Dieser Wert wird auch zur maximalen Jog-Geschwindigkeit.

> **NOTE** Der Standardwert ist 600, falls nicht festgelegt.

Makro-Buttons

Sie können Makros zu GMOCCAPY hinzufügen, ähnlich wie bei Touchy. Ein Makro ist nichts anderes als eine NGC-Datei. Sie können komplette CNC-Programme im MDI-Modus ausführen, indem Sie nur einen Knopf drücken. Dazu müssen Sie einen Suchpfad für Makros hinzufügen:

```
[RS274NGC]
SUBROUTINE_PATH = macros
```

Hier wird der Pfad für die Suche nach Makros und anderen Unterprogrammen festgelegt. Mehrere Unterprogramm-Pfade können mit ":" getrennt werden.

Dann müssen Sie nur noch einen Abschnitt wie diesen hinzufügen:

Konfiguration von fünf Makros, die in der MDI-Buttonliste angezeigt werden sollen

```
[MACROS]
MACRO = i_am_lost
MACRO = hello_world
MACRO = jog_around
MACRO = increment xinc yinc
MACRO = go_to_position X-pos Y-pos Z-pos
```

Dann müssen Sie die entsprechenden NGC-Dateien bereitstellen, die diesen Regeln entsprechen müssen:

- Der Name der Datei muss genau dem in der Makrozeile angegebenen Namen entsprechen, nur mit der Erweiterung ".ngc" (Groß- und Kleinschreibung beachten).
- Die Datei muss ein Unterprogramm enthalten: ***O<i_am_lost> sub***, der Name des Unterprogramms muss genau (Groß-/Kleinschreibung beachten) mit dem Namen des Makros übereinstimmen.

- Die Datei muss mit einem endsub ***O<i_am_lost> endsub*** gefolgt von einem ***M2***-Befehl enden.
- Die Dateien müssen in einem Ordner abgelegt werden, der in Ihrer INI-Datei durch *SUBROUTINE_PATH* im Abschnitt RS274NGC angegeben ist

Der Code zwischen sub und endsub wird durch Betätigen der entsprechenden Makrotaste ausgeführt.

NOTE Es werden maximal 16 Makros in der GUI angezeigt. Aus Platzgründen kann es sein, dass Sie auf einen Pfeil klicken müssen, um die Seite zu wechseln und versteckte Makroschaltflächen anzuzeigen. Die Makro-Schaltflächen werden in der Reihenfolge der INI-Einträge angezeigt. Es ist kein Fehler, mehr als 16 Makros in Ihrer INI-Datei zu platzieren, sie werden nur nicht angezeigt.

NOTE Sie finden die Beispielmakros in einem Ordner mit dem Namen *macros*, der sich im GMOCCAPY sim-Ordner befindet. Wenn Sie mehrere Pfade für Unterprogramme angegeben haben, werden diese in der Reihenfolge der angegebenen Pfade durchsucht. Die erste gefundene Datei wird verwendet.

GMOCCAPY akzeptiert auch Makros, die nach Parametern wie den folgenden fragen:

```
[MACROS]
MACRO = go_to_position X-pos Y-pos Z-pos
```

Die Parameter müssen durch Leerzeichen getrennt werden. Dieses Beiaspiel ruft eine Datei "go_to_position.ngc" mit dem folgenden Inhalt auf:

```
; Testdatei "go to position" (engl. für geh' zur Position)
; fährt die Maschine zu einer bestimmten Position

O<go_to_position> sub

G17
G21
G54
G61
G40
G49
G80
G90

;#1 = <X-Pos>
;#2 = <Y-Pos>
;#3 = <Z-Pos>

(DEBUG, wird jetzt die Maschine zu X = #1 , Y = #2 , Z = #3 bewegen)
G0 X #1 Y #2 Z #3

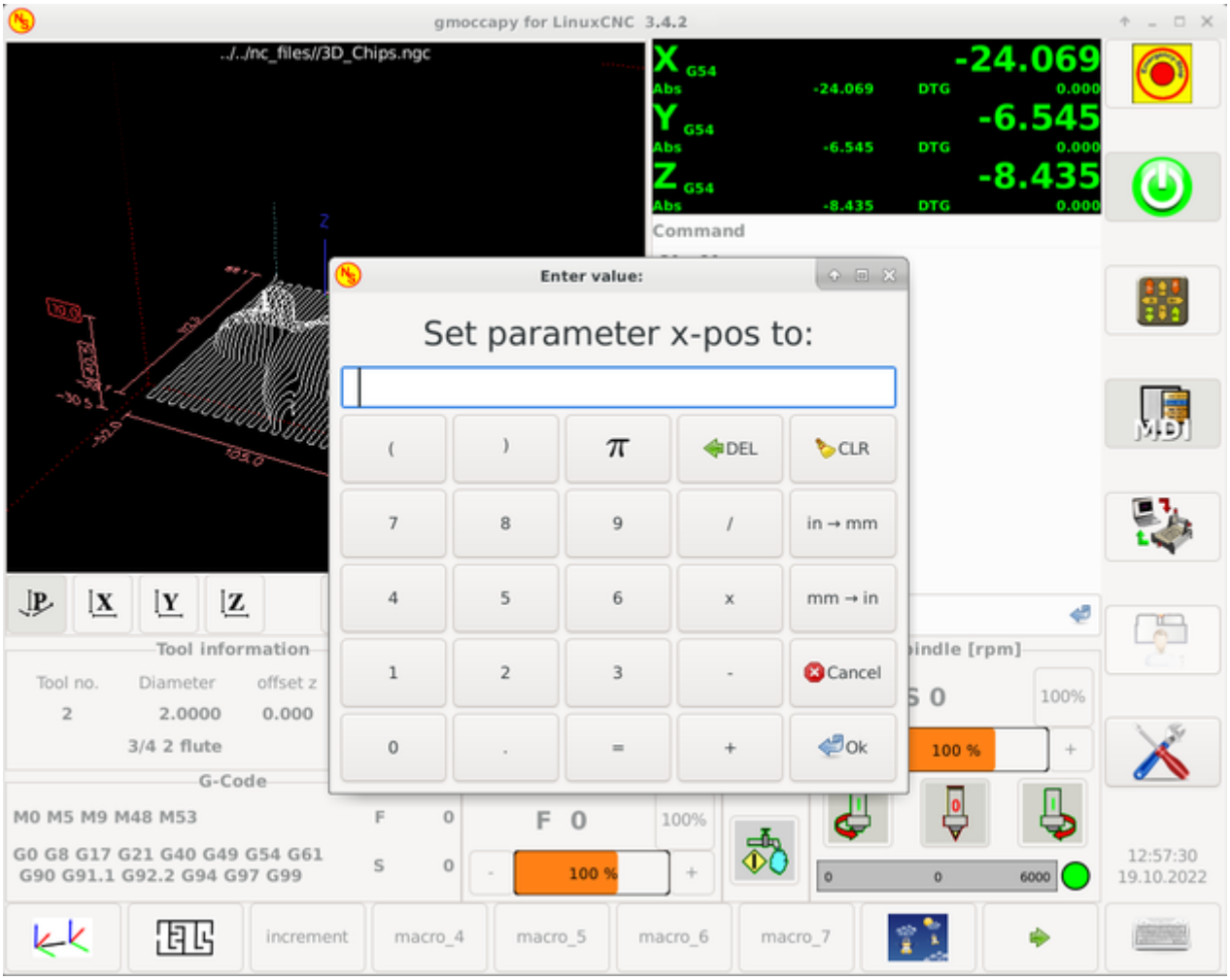
O<go_to_position> endsub
M2
```

Nach dem Drücken der Taste ***Makro ausführen*** werden Sie aufgefordert, die Werte für ***X-pos Y-pos Z-***

pos einzugeben, und das Makro wird nur ausgeführt, wenn alle Werte angegeben wurden.

NOTE Wenn Sie ein Makro ohne Bewegung verwenden möchten, beachten Sie auch die Hinweise in bekannte Probleme.

Makrobeispiel mit dem "Gehe zu Position"-Makro



Eingebettete Registerkarten und Bedienfelder

Sie können GMOCCAPY eingebettete Programme hinzufügen, wie Sie es bei AXIS, Touchy und Gscreen tun können. Alles wird von GMOCCAPY automatisch erledigt, wenn Sie ein paar Zeilen in Ihre INI-Datei im DISPLAY-Abschnitt einfügen.

If you have never used a Glade panel, I recommend to read the excellent documentation on Glade VCP. You can embed QtVCP panels too: QtVCP panels

Beispiel für eingebettete Registerkarten

```
EMBED_TAB_NAME = DRO
EMBED_TAB_LOCATION = ntb_user_tabs
EMBED_TAB_COMMAND = gladevcp -x {XID} dro.glade

EMBED_TAB_NAME = Second user tab
EMBED_TAB_LOCATION = ntb_preview
```

```
EMBED_TAB_COMMAND = gladevcp -x {XID} vcp_box.glade
```

Alles, was Sie beachten müssen, ist, dass Sie für jede Registerkarte oder jedes Seitenfeld die genannten drei Zeilen einfügen:

- EMBED_TAB_NAME = Stellt den Namen der Registerkarte oder des Seitenfensters dar, es ist Ihnen überlassen, welchen Namen Sie verwenden, aber er muss vorhanden sein!
- EMBED_TAB_LOCATION = Der Ort, an dem Ihr Programm in der GUI platziert wird, siehe Abbildung Embedded tab locations. Gültige Werte sind:
  - **ntb_user_tabs** (als Hauptregisterkarte, die den gesamten Bildschirm abdeckt)
  - **ntb_preview** (als Tab auf der Vorschauseite **(1)**)
  - **hbox_jog** (blendet die Jog-Buttons aus und führt die Glade-Datei hier ein **(2)**)
  - **box_left** (auf der linken Seite, ganz oben auf dem Bildschirm)
  - **box_right** (auf der rechten Seite, zwischen dem normalen Bildschirm und der Schaltflächenliste **(11)**)
  - **box_tool_and_code_info** (blendet die Werkzeuginformationen und die G-Code-Frames aus und fügt Ihre Glade-Datei hier ein **(3)**)
  - **box_tool_info** (blendet den Werkzeuginformationsrahmen aus und fügt hier Ihre Glade-Datei ein **(3a)**)
  - **box_code_info** (blendet den G-Code-Informationsrahmen aus und fügt hier Ihre Glade-Datei ein **(3b)**)
  - **box_vel_info** (blendet die Geschwindigkeitsrahmen aus und fügt Ihre Glade-Datei ein **(4)**)
  - **box_coolant_and_spindle** (blendet die Rahmen für Kühlmittel und Spindel aus und fügt die Glade-Datei hier ein **(5)**+**(6)**)
  - **box_cooling** (blendet den Kühlrahmen aus und fügt Ihre Glade-Datei ein **(5)**)
  - **box_spindle** (blendet den Spindelrahmen aus und fügt Ihre Glade-Datei ein **(6)**)
  - **box_custom_1** (fügt Ihre Glade-Datei links von vel_frame ein)
  - **box_custom_2** (fügt Ihre Glade-Datei links von cooling_frame ein)
  - **box_custom_3** (fügt Ihre Glade-Datei links von spindle_frame ein)
  - **box_custom_4** (fügt Ihr Glade-Datei rechts vom spindle_frame ein)
  - **box_dro_side** (führt die Glade-Datei rechts vom DRO ein)
  - **ntb_preview** (als Tab auf der Vorschauseite **(1)**)

NOTE Siehe auch die mitgelieferten INI-Beispieldateien, um die Unterschiede zu sehen.

- EMBED_TAB_COMMAND = Der auszuführende Befehl, d. h.

```
gladevcp -x {XID} dro.glade
```

enthält eine benutzerdefinierte Glade-Datei mit dem Namen dro.glade an dem genannten Ort. Die

Datei muss sich im Konfigurationsordner Ihres Computers befinden.

```
gladevcp h_buttonlist.glade
```

öffnet einfach ein neues Benutzerfenster mit dem Namen h_buttonlist.glade. Beachten Sie den Unterschied, dass dieses Fenster eigenständig ist und unabhängig vom GMOCCAPY-Fenster verschoben werden kann.

```
gladevcp -c gladevcp -u hitcounter.py -H manual-example.hal manual-example.ui
```

fügt das Panel manual-example.ui hinzu, fügt einen benutzerdefinierten Python-Handler, hitcounter.py, ein und stellt alle Verbindungen her, nachdem das Panel gemäß manual-example.hal realisiert wurde.

```
hide (engl. für ausblenden)
```

blendet das gewählte Kästchen aus.

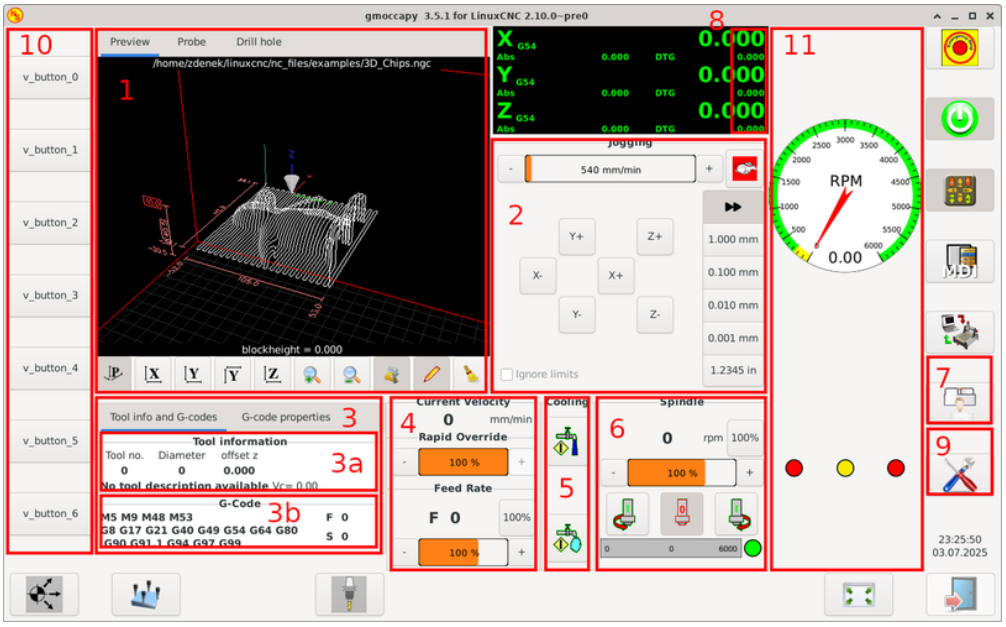


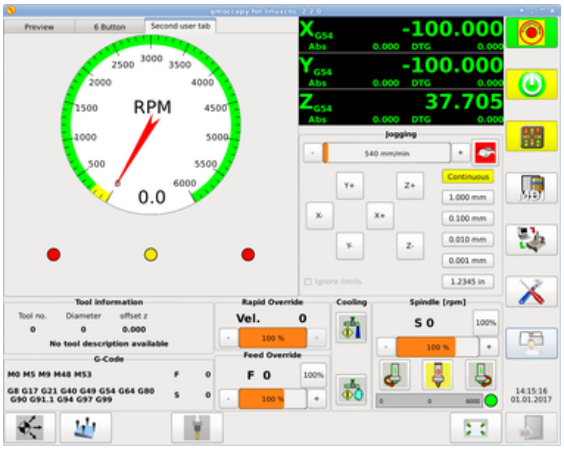
Figure 162. Eingebettete Registerkartenpositionen

NOTE

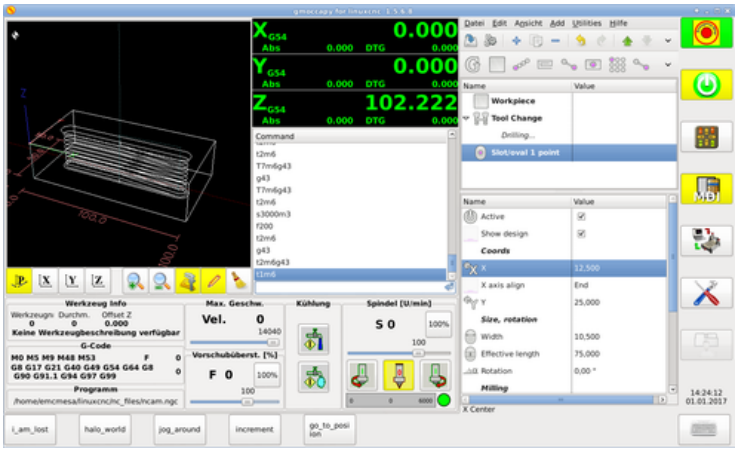
Wenn Sie HAL-Verbindungen zu Ihrem benutzerdefinierten Glade-Panel herstellen, müssen Sie dies in der HAL-Datei tun, die in der EMBED_TAB_COMMAND-Zeile angegeben ist, andernfalls erhalten Sie möglicherweise die Fehlermeldung, dass der HAL-Pin nicht existiert - dies ist auf Race Conditions beim Laden der HAL-Dateien zurückzuführen. Verbindungen zu GMOCCAPY HAL-Pins müssen in der postgui-HAL-Datei hergestellt werden, die in Ihrer INI-Datei angegeben ist, da diese Pins vor der Realisierung der GUI nicht existieren.

Hier sind einige Beispiele:

ntb_preview



box_right - and GMOCCAPY in MDI mode



Benutzerdefinierte Mitteilungen

GMOCCAPY hat die Möglichkeit, HAL-gesteuerte Benutzermeldungen zu erstellen. Um sie zu verwenden, müssen Sie einige Zeilen in den [DISPLAY]-Abschnitt der INI-Datei einfügen.

Diese drei Zeilen werden benötigt, um einen Popup-Meldungsdialog für den Benutzer zu definieren:

```
MESSAGE_TEXT = Der anzuzeigende Text, kann mit Tango-Markup formatiert sein
MESSAGE_TYPE = "status" , "okdialog" , "yesnodialog"
MESSAGE_PINNAME = ist der Name der zu erstellenden HAL-Pin-Gruppe
```

Die Nachrichten unterstützen die Auszeichnungssprache Pango. Detaillierte Informationen über die Auszeichnungssprache finden Sie unter Pango Markup.

Die folgenden drei Dialogtypen sind verfügbar:

- **status** - Zeigt eine Nachricht in einem Pop-up-Fenster an, das Nachrichtensystem System von GMOCCAPY nutzend.
- **okdialog** - Hält den Fokus auf dem Nachrichtendialog und aktiviert einen `-waiting`(engl. für wartenden) HAL-Pin.
- **yesnodialog** - Hält den Fokus auf dem Nachrichtendialog und aktiviert einen `-waiting` (engl. für wartenden) HAL-Pin und einen `-response` (engl. für Antwort) HAL-Pin.

Ausführlichere Informationen zu den Pins finden Sie unter Von Benutzer erstelle HAL Pins für Nachrichten.

Beispiel für die Konfiguration von Benutzernachrichten

```
MESSAGE_TEXT = This is a <span background="#ff0000" foreground="#ffffff">info-
message</span> test
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest

MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yesnodialog
```

```
MESSAGE_TEXT = Text can be <small>small</small>, <big>big</big>, <b>bold</b
<i>italic</i>, and even be <span color="red">colored</span>.
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = okdialog
```

NOTE Derzeit funktioniert die Formatierung nicht.

Vorschau Kontrolle

Magische Kommentare können verwendet werden, um die G-Code-Vorschau zu steuern. Bei sehr großen Programmen kann die Vorschau sehr lange zum Laden brauchen. Sie können steuern, was auf dem Grafikbildschirm angezeigt und was ausgeblendet wird, indem Sie die entsprechenden Kommentare aus dieser Liste in Ihren G-Code einfügen:

```
(PREVIEW,hide)
<zu versteckender G-code>
(PREVIEW,show)
```

Benutzerdefinierte Befehlsdatei

Wenn eine Datei `~/.gmoccapyrc` existiert, wird ihr Inhalt als Python-Quellcode ausgeführt, kurz nach der Anzeige der grafischen Benutzeroberfläche. Die Einzelheiten dessen, was in `~/.gmoccapyrc` geschrieben werden kann, können sich während des Entwicklungszyklus ändern.

Eine konfigurationsspezifische Python-Datei kann in der INI-Datei angegeben werden

```
[DISPLAY]
USER_COMMAND_FILE=dateiname.py
```

Ist diese Datei angegeben, so wird diese Datei direkt nach der Anzeige der GMOCCAPY-GUI **anstelle** von `~/.gmoccapyrc` aufgerufen.

Im folgenden Beispiel wird die Größe der vertikalen Schaltflächen geändert: .Beispiel für .gmoccapyrc Datei

```
self.widgets.vbtb_main.set_size_request(85,-1)
BB_SIZE = (70, 70) # default = (90, 56)
self.widgets.tbtn_estop.set_size_request(*BB_SIZE)
self.widgets.tbtn_on.set_size_request(*BB_SIZE)
self.widgets.rbt_manual.set_size_request(*BB_SIZE)
self.widgets.rbt_mdi.set_size_request(*BB_SIZE)
self.widgets.rbt_auto.set_size_request(*BB_SIZE)
self.widgets.tbtn_setup.set_size_request(*BB_SIZE)
self.widgets.tbtn_user_tabs.set_size_request(*BB_SIZE)
self.widgets.btn_exit.set_size_request(*BB_SIZE)
```

Die Widget-Namen können in der Datei /usr/share/gmoccapy.glade nachgeschlagen werden

Benutzerdefinierte CSS-Datei

Ähnlich wie bei der Benutzerbefehlsdatei (engl. user command file) ist es möglich, das Erscheinungsbild durch Cascading Style Sheets (CSS) zu beeinflussen. Existiert eine Datei `~/.gmoccapy_css`, wird deren Inhalt in den Stylesheet-Provider geladen und so auf die GUI angewendet.

Eine konfigurationsspezifische CSS-Datei kann mit einer INI-Datei-Einstellung angegeben werden

```
[DISPLAY]
USER_CSS_FILE=dateiname.css
```

Ist diese Datei angegeben, so wird diese Datei **anstelle** von `~/.gmoccapy_css` verwendet.

Informationen, was mit CSS gesteuert werden kann, finden Sie hier: Übersicht über CSS in GTK

Hier ein Beispiel dafür, wie die Farbe der markierten Schaltflächen auf gelb gesetzt werden kann: .Beispiel mit gelber Farbe für markierte Checkbuttons

```
button:checked {
    background: rgba(250,230,0,0.8);
}
```

Protokollierung (engl. logging)

GMOCCAPY unterstützt die Angabe der Informationsebene (Loglevel), die auf der Konsole und in der Logdatei ausgegeben wird.

Die Reihenfolge ist *VERBOSE*, *DEBUG*, *INFO*, *WARNING*, *ERROR*, *CRITICAL*. Standard ist *WARNING*, d.h. *WARNING*, *ERROR* und *CRITICAL* werden gedruckt.

Sie können die Protokollstufe in der INI-Datei wie folgt angeben:

```
[DISPLAY]
DISPLAY = gmoccapy <log_level_param>
```

unter Verwendung dieser Parameter:

```
Log level   <log_level_param>
DEBUG       -d
INFO        -i
VERBOSE     -v
ERROR       -q
```

Beispiel: Konfigurieren der Protokollierung, um nur Fehler zu drucken

```
[DISPLAY]
DISPLAY = gmoccapy -q
```

Sie können angeben, wo die Log-Datei gespeichert werden soll:

```
[DISPLAY]
LOG_FILE = gmoccapy.log
```

Wenn LOG_FILE nicht festgelegt ist, erfolgt die Protokollierung in $HOME/<base_log_name>.log.

10.2.5. HAL-Pins

GMOCCAPY exportiert mehrere HAL-Pins, um auf Hardware-Geräte reagieren zu können. Das Ziel ist es, eine GUI zu erhalten, die in einem Werkzeugbau komplett/meist ohne Maus oder Tastatur bedient werden kann.

NOTE

Sie müssen alle Verbindungen zu GMOCCAPY-Pins in Ihrer postgui.hal-Datei herstellen. Wenn GMOCCAPY gestartet wird, erstellt es die HAL-Pins für die GUI und führt dann die in der INI-Datei genannte post-GUI-HAL-Datei aus:

```
[HAL]
POSTGUI_HALFILE=<Dateiname>
```

Typischerweise wäre <Dateiname> der Basisname der Konfiguration + _postgui.hal, z.B. lathe_postgui.hal, aber es kann jeder legale Dateiname sein.
Diese Befehle werden nach der Erstellung des Bildschirms ausgeführt und garantieren, dass die HAL-Pins des Widgets verfügbar sind.
Sie können mehrere Zeilen mit POSTGUI_HALFILE=<Dateiname> in der INI-Datei haben. Sie werden nacheinander in der Reihenfolge ausgeführt, in der sie erscheinen.

Listen mit rechten und unteren Schaltflächen

Der Bildschirm hat zwei Haupt-Button-Leisten, eine auf der rechten Seite und eine am unteren Rand. Die Tasten auf der rechten Seite ändern sich während des Betriebs nicht, aber die untere Button-Leiste ändert sich sehr oft. Die Buttons werden von oben nach unten und von links nach rechts gezählt, beginnend mit 0.

NOTE

Die Namen der Pins wurden in GMOCCAPY 2 geändert, um sie besser anordnen zu können.

Die Pins für die rechten (vertikalen) Buttons sind:

- **gmoccapy.v-button.button-0** *(bool IN)*
- **gmoccapy.v-button.button-1** *(bool IN)*
- **gmoccapy.v-button.button-2** *(bool IN)*
- **gmoccapy.v-button.button-3** *(bool IN)*
- **gmoccapy.v-button.button-4** *(bool IN)*
- **gmoccapy.v-button.button-5** *(bool IN)*
- **gmoccapy.v-button.button-6** *(bool IN)*

Für die unteren (horizontalen) Buttons sind dies:

- **gmoccapy.h-button.button-0** *(bool IN)*
- **gmoccapy.h-button.button-1** *(bool IN)*
- **gmoccapy.h-button.button-2** *(bool IN)*
- **gmoccapy.h-button.button-3** *(bool IN)*
- **gmoccapy.h-button.button-4** *(bool IN)*
- **gmoccapy.h-button.button-5** *(bool IN)*
- **gmoccapy.h-button.button-6** *(bool IN)*
- **gmoccapy.h-button.button-7** *(bool IN)*
- **gmoccapy.h-button.button-8** *(bool IN)*
- **gmoccapy.h-button.button-9** *(bool IN)*

Da sich die Tasten in der unteren Liste je nach Modus und anderen Einflüssen ändern, aktivieren die Hardware-Tasten die angezeigten Funktionen. Sie müssen sich also nicht um das Umschalten von Funktionen in HAL kümmern, denn das übernimmt GMOCCAPY komplett!

Bei einer dreiachsigen XYZ-Maschine reagieren die HAL-Pins wie in den folgenden drei Tabellen dargestellt:

Table 57. Funktionelle Zuordnung der horizontalen Buttons (1)

Pin	Manueller Modus	MDI Modus	Auto-Modus
gmoccapy.h-button.button-0	Knöpfe für Referenzfahrt anzeigen	macro 1 (wenn definiert)	Datei öffnen
gmoccapy.h-button.button-1	Knöpfe zum Antasten anzeigen	macro 2 (wenn definiert)	Programm neu laden
gmoccapy.h-button.button-2		macro 3 (wenn definiert)	ausführen
gmoccapy.h-button.button-3	Werkzeugverwaltung anzeigen	macro 4 (wenn definiert)	stop
gmoccapy.h-button.button-4		macro 5 (wenn definiert)	pausiere
gmoccapy.h-button.button-5		macro 6 (wenn definiert)	schrittweise
gmoccapy.h-button.button-6		macro 7 (wenn definiert)	aus der Zeile ausführen, sofern in den Einstellungen aktiviert, sonst nichts
gmoccapy.h-button.button-7		macro 8 (wenn definiert)	optionale Blöcke

Pin	Manueller Modus	MDI Modus	Auto-Modus
gmoccapy.h-button.button-8	Vorschau in voller Größe	macro 9 oder Button um zusätzliche Makros anzuzeigen	Vorschau in voller Größe
gmoccapy.h-button.button-9	exit wenn die Maschine ausgeschaltet ist, sonst nichts	Tastatur öffnen oder abbrechen, wenn das Makro läuft	Code bearbeiten

Table 58. Funktionszuweisung der horizontalen Buttons (2)

Pin	Einstellungs-Modus	Homing-Modus	Touch-Off-Modus
gmoccapy.h-button.button-0	MDI-Verlauf löschen		touch X
gmoccapy.h-button.button-1		Referenzfahrt aller Achsen	touch Y
gmoccapy.h-button.button-2			touch Z
gmoccapy.h-button.button-3		Referenzfahrt X	
gmoccapy.h-button.button-4	Öffnen von ClassicLadder	Referenzfahrt Y	
gmoccapy.h-button.button-5	öffnen von HAL-scope	Referenzfahrt Z	
gmoccapy.h-button.button-6	HAL-Status öffnen		
gmoccapy.h-button.button-7	HAL-Meter öffnen		
gmoccapy.h-button.button-8	HAL-Kalibrierung öffnen	unhome all	
gmoccapy.h-button.button-9	öffnen von HAL-show	zurück (engl. back)	zurück (engl. back)

Table 59. Funktionelle Zuordnung der horizontalen Buttons (3)

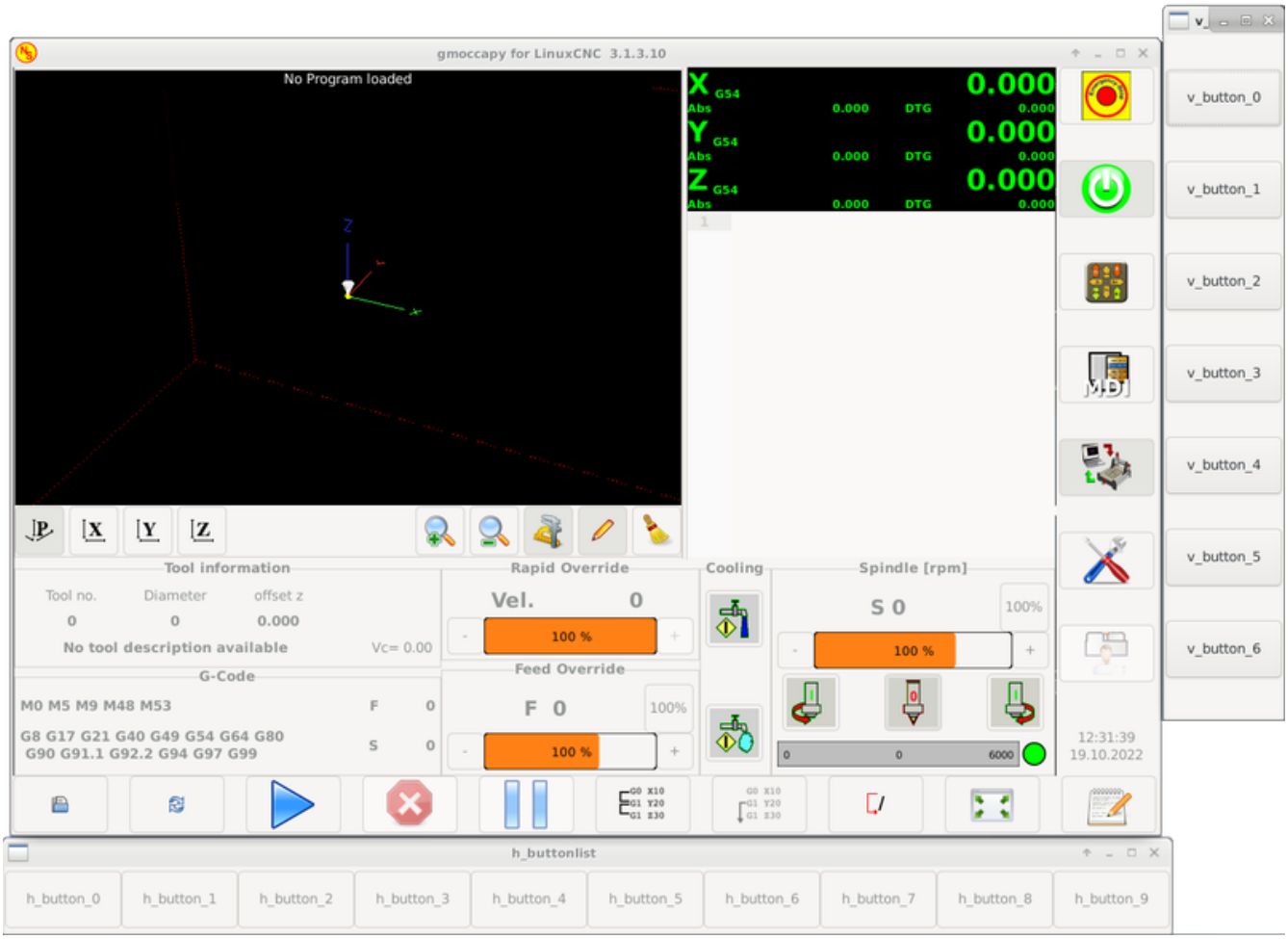
Pin	Werkzeug-Modus	Bearbeitungsmodus	Datei auswählen
gmoccapy.h-button.button-0			Wechsel in das Home-Verzeichnis
gmoccapy.h-button.button-1		Datei neu laden	ein Verzeichnis nach oben

Pin	Werkzeug-Modus	Bearbeitungsmodus	Datei auswählen
gmoccapy.h-button.button-2		speichern	
gmoccapy.h-button.button-3		speichern unter	Auswahl nach links verschieben
gmoccapy.h-button.button-4	Werkzeugwechsel mit Nummer T? M6		Auswahl nach rechts verschieben
gmoccapy.h-button.button-5	Werkzeug nach Nummer ohne Wechsel einstellen M61 Q?		Sprung in das in den Einstellungen festgelegte Verzeichnis
gmoccapy.h-button.button-6	Wechsel zum ausgewählten Werkzeug	Neue Datei	
gmoccapy.h-button.button-7			auswählen / EINGABETASTE
gmoccapy.h-button.button-8	Werkzeugberührung in Z	Tastatur zeigen	
gmoccapy.h-button.button-9	zurück (engl. back)	zurück (engl. back)	zurück (engl. back)

Wir haben also 67 Reaktionen mit nur 10 HAL-Pins!

Diese Pins werden zur Verfügung gestellt, um den Bildschirm ohne Touchpanel verwenden zu können, oder um ihn vor übermäßiger Benutzung zu schützen, indem Hardware-Tasten um das Panel herum platziert werden. Sie sind in einer Beispielkonfiguration verfügbar, wie im Bild unten gezeigt.

Beispielkonfiguration "gmoccapy_sim_hardware_button" mit den seitlichen Schaltflächen



Geschwindigkeiten Übersteuerungen (overrides)

Alle Schieberegler von GMOCCAPY können mit Hardware-Encodern oder Hardware-Potentiometern verbunden werden.

NOTE Für GMOCCAPY 3 haben sich einige HAL-Pin-Namen geändert, als neue Kontrollen implementiert wurden. Die maximale Geschwindigkeit existiert nicht mehr, sie wurde aufgrund der Nachfrage vieler Benutzer durch eine schnelle Übersteuerung ersetzt.

Um *Encoder* anzuschließen, werden folgende Pin exportiert:

- **gmoccapy.jog.jog-velocity.counts** *(sint IN)* - Jog velocity
- **gmoccapy.jog.jog-velocity.count-enable** *(bool IN)* - Must be True, to enable counts
- **gmoccapy.feed.feed-override.counts** *(sint IN)* - feed override
- **gmoccapy.feed.feed-override.count-enable** *(bool IN)* - Must be True, to enable counts
- **gmoccapy.feed.reset-feed-override** *(bool IN)* - reset the feed override to 0%
- **gmoccapy.spindle.spindle-override.counts** *(sint IN)* - spindle override
- **gmoccapy.spindle.spindle-override.count-enable** *(bool IN)* - Must be True, to enable counts
- **gmoccapy.spindle.reset-spindle-override** *(bool IN)* - reset the spindle override to 0%
- **gmoccapy.rapid.rapid-override.counts** *(sint IN)* - Maximal velocity of the machine

- **gmoccapy.rapid.rapid-override.count-enable** *(bool IN)* - Must be True, to enable counts

Um Potentiometer anzuschließen, verwenden Sie die folgenden Pins:

- **gmoccapy.jog.jog-velocity.direct-value** *(real IN)* - To adjust the jog velocity slider
- **gmoccapy.jog.jog-velocity.analog-enable** *(bool IN)* - Must be True, to allow analog inputs
- **gmoccapy.feed.feed-override.direct-value** *(real IN)* - To adjust the feed override slider
- **gmoccapy.feed.feed-override.analog-enable** *(bool IN)* - Must be True, to allow analog inputs
- **gmoccapy.spindle.spindle-override.direct-value** *(real IN)* - To adjust the spindle override slider
- **gmoccapy.spindle.spindle-override.analog-enable** *(bool IN)* - Must be True, to allow analog inputs
- **gmoccapy.rapid.rapid-override.direct-value** *(real IN)* - To adjust the max velocity slider
- **gmoccapy.rapid.rapid-override.analog-enable** *(bool IN)* - Must be True, to allow analog inputs

Darüber hinaus bietet GMOCCAPY 3 zusätzliche HAL-Pins zur Steuerung der neuen Schieberegler-Widgets mit Taster. Die Werte, wie schnell der Anstieg oder Abfall sein soll, müssen in der Glade-Datei eingestellt werden. In einer zukünftigen Version wird dies in die Einstellungsseite integriert werden.

GESCHWINDIGKEIT

- **gmoccapy.spc_jog_vel.increase** *(bool IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_jog_vel.decrease** *(bool IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_jog_vel.scale** *(real IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_jog_vel.value** *(real OUT)* - Value of the widget
- **gmoccapy.spc_jog_vel.scaled-value** *(real OUT)* - Scaled value of the widget

FEED

- **gmoccapy.spc_feed.increase** *(bool IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_feed.decrease** *(bool IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_feed.scale** *(real IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_feed.value** *(real OUT)* - Value of the widget
- **gmoccapy.spc_feed.scaled-value** *(real OUT)* - Scaled value of the widget

SPINDLE

- **gmoccapy.spc_spindle.increase** *(bool IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_spindle.decrease** *(bool IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_spindle.scale** *(real IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_spindle.value** *(real OUT)* - Value of the widget
- **gmoccapy.spc_spindle.scaled-value** *(real OUT)* - Scaled value of the widget

RAPIDS

- **gmoccapy.spc_rapid.increase** *(bool IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_rapid.decrease** *(bool IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_rapid.scale** *(real IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_rapid.value** *(real OUT)* - Value of the widget
- **gmoccapy.spc_rapid.scaled-value** *(real OUT)* - Scaled value of the widget

The real pins do accept values from 0.0 to 1.0, being the percentage value you want to set the slider value.

WARNING	Wenn Sie beide Anschlussarten verwenden, schließen Sie nicht denselben Schieberegler an beide Pins an, da die Einflüsse zwischen den beiden nicht getestet wurden! Verschiedene Schieberegler können an die eine oder andere HAL-Anschlussart angeschlossen werden.
IMPORTANT	Bitte beachten Sie, dass die Jog-Geschwindigkeit vom Zustand der Schildkrötentaste abhängt und je nach Modus (Schildkröte oder Kaninchen) zu unterschiedlichen Schiebereglern führt. Bitte werfen Sie auch einen Blick auf den Abschnitt zu Jog Geschwindigkeiten und der Turtle-Jog HAL Pin für weitere Details.

Example 4. Einstellen eines Schiebereglerwerts

```
Spindle Override Min Value = 20 %
Spindle Override Max Value = 120 %
gmoccapy.analog-enable = 1
gmoccapy.spindle-override-value = 0.25

value to set = Min Value + (Max Value - Min Value) * gmoccapy.spindle-override-value
value to set = 20 + (120 - 20) * 0.25
value to set = 45 %
```

Jog HAL Pins

Alle Achsen, die in der INI-Datei angegeben sind, haben einen Jog-Plus- und einen Jog-Minus-Pin, so dass Hardware-Taster verwendet werden können, um die Achse zu joggen.

NOTE	Die Benennung dieser HAL-Pins hat sich in GMOCCAPY 2 geändert.

Für die Standard-XYZ-Konfiguration sind folgende HAL-Pins verfügbar:

- **gmoccapy.jog.axis.jog-x-plus** *(bool IN)*
- **gmoccapy.jog.axis.jog-x-minus** *(bool IN)*

- **gmoccapy.jog.axis.jog-y-plus** *(bool IN)*
- **gmoccapy.jog.axis.jog-y-minus** *(bool IN)*
- **gmoccapy.jog.axis.jog-z-plus** *(bool IN)*
- **gmoccapy.jog.axis.jog-z-minus** *(bool IN)*

Wenn Sie eine 4-Achsen-Konfiguration verwenden, gibt es zwei zusätzliche Pins:

- **gmoccapy.jog.jog-<your fourth axis letter >-plus** *(bool IN)*
- **gmoccapy.jog.jog-<your fourth axis letter >-minus** *(bool IN)*

Für eine C-Achse werden Sie sehen:

- **gmoccapy.jog.axis.jog-c-plus** *(bool IN)*
- **gmoccapy.jog.axis.jog-c-minus** *(bool IN)*

Jog-Geschwindigkeiten und Turtle-Jog HAL-Pin

Die Tippgeschwindigkeit kann mit dem entsprechenden Schieberegler ausgewählt werden. Die Skala des Schiebereglers wird geändert, wenn der Schildkrötenknopf (derjenige, der einen Hasen oder eine Schildkröte zeigt) umgeschaltet wurde. Wenn die Schaltfläche nicht sichtbar ist, wurde sie möglicherweise auf der Seite Einstellungen deaktiviert. Wenn die Schaltfläche das Kaninchen-Symbol zeigt, ist die Skala von der minimalen bis zur maximalen Geschwindigkeit der Maschine. Zeigt sie die Schildkröte, erreicht die Skala standardmäßig nur 1/20 der maximalen Geschwindigkeit. Der verwendete Teiler kann auf der Einstellungsseite eingestellt werden.

Mit einem Touchscreen ist es also viel einfacher, kleinere Geschwindigkeiten auszuwählen.

GMOCCAPY bietet diesen HAL-Pin an, um zwischen Hasen- und Igel-Joggen zu wechseln:

- **gmoccapy.jog.turtle-jog** *(bool IN)*

Jog-Schrittweiten HAL-Pins

Die in der INI-Datei angegebenen Jog-Inkremente wie

```
[DISPLAY]
INCREMENTS = 5mm 1mm .5mm .1mm .05mm .01mm
```

sind über HAL-Pins auswählbar, so dass ein Hardware-Schalter verwendet werden kann, um die zu verwendende Schrittweite auszuwählen. Es gibt maximal 10 HAL-Pins für die in der INI-Datei angegebenen Inkremente. Wenn Sie mehr Inkremente in Ihrer INI-Datei angeben, sind diese nicht über die GUI erreichbar, da sie nicht angezeigt werden.

Wenn Sie 6 Inkremente in Ihrer INI-Datei haben, wie im obigen Beispiel, erhalten Sie 7 Pins:

- **gmoccapy.jog.jog-inc-0** *(bool IN)* - This one is fixed and will represent continuous jogging.
- **gmoccapy.jog.jog-inc-1** *(bool IN)* - First increment given in the INI file.

- **gmoccapy.jog.jog-inc-2** *(bool IN)*
- **gmoccapy.jog.jog-inc-3** *(bool IN)*
- **gmoccapy.jog.jog-inc-4** *(bool IN)*
- **gmoccapy.jog.jog-inc-5** *(bool IN)*
- **gmoccapy.jog.jog-inc-6** *(bool IN)*

GMOCCAPY bietet auch einen HAL-Pin zur Ausgabe der gewählten Jog-Schrittweite:

- **gmoccapy.jog.jog-increment** *(real OUT)*

Hardware-Entsperr-Pin

Um einen Schlüsselschalter zum Entsperren der Einstellungsseite verwenden zu können, wird der folgende Pin exportiert:

- **gmoccapy.unlock-settings** *(bool IN)* - The settings page is unlocked if the pin is high. To use this pin, you need to activate it on the settings page.

Fehler/Warnung Pins

- **gmoccapy.error** *(bool OUT)* - Indicates an error, so a light can lit or even the machine may be stopped. It will be reset with the pin `gmoccapy.delete-message`.
- **gmoccapy.delete-message** *(bool IN)* - Will delete the first error and reset the `gmoccapy.error` pin to false after the last error has been cleared.
- **gmoccapy.warning-confirm** *(bool IN)* - Confirms warning dialog like click on OK

NOTE	Meldungen oder Benutzerinformationen haben keinen Einfluss auf den `gmoccapy.error`-Pin, aber der `gmoccapy.delete-message`-Pin löscht die letzte Meldung, wenn kein Fehler angezeigt wird!

Benutzerdefinierte HAL-Pins für Mitteilungen

GMOCCAPY kann so konfiguriert werden, dass es auf externe Fehler mit 3 verschiedenen Benutzermeldungen reagiert:

status

- **gmoccapy.messages.status** *(bool IN)* - Triggers the dialog.

okdialog

- **gmoccapy.messages.okdialog** *(bool IN)* - Triggers the dialog.
- **gmoccapy.messages.okdialog-waiting** *(bool OUT)* - Will be *1* as long as the dialog is open. Closing the message will reset the this pin.

yesnodialog

- **gmoccapy.messages.yesnodialog** *(bool IN)* - Triggers the dialog.

- **gmoccapy.messages.yesnodialog-waiting** *(bool OUT)* - Will be *1* as long as the dialog is open. Closing the message will reset the this pin.
- **gmoccapy.messages.yesnodialog-response** *(bool OUT)* - This pin will change to *1* if the user clicks OK and in all other cases it will be *0*. This pin will remain *1* until the dialog is called again.

Um eine vom Benutzer erstellte Nachricht hinzuzufügen, müssen Sie die Nachricht zur INI-Datei im Abschnitt DISPLAY hinzufügen. Siehe Konfiguration von benutzererstellten Nachrichten.

Beispiel für eine Benutzermeldung (INI-Datei)

```
MESSAGE_TEXT = LUBE FAULT
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = lube-fault

MESSAGE_TEXT = X SHEAR PIN BROKEN
MESSAGE_TYPE = status
MESSAGE_PINNAME = pin
```

Um diese neuen Pins zu verbinden, müssen Sie dies in der postgui-HAL-Datei tun. Hier sind einige Beispielverbindungen zum Verbinden der Nachrichtensignale mit anderen Stellen in der HAL-Datei.

Beispiel für den Anschluss von Benutzernachrichten (HAL-Datei)

```
net gmoccapy-lube-fault gmoccapy.messages.lube-fault
net gmoccapy-lube-fault-waiting gmoccapy.messages.lube-fault-waiting
net gmoccapy-pin gmoccapy.messages.pin
```

Für weitere Informationen über HAL-Dateien und den net-Befehl siehe HAL Basics.

Spindel-Feedback-Pins

Es gibt zwei Pins für das Spindelfeedback:

- **gmoccapy.spindle_feedback_bar** *(real IN)* - Pin to show the spindle speed on the spindle bar.
- **gmoccapy.spindle_at_speed_led** *(bool IN)* - Pin to lit the is-at-speed-led.

Pins zur Anzeige vom Programmfortschritt

There are six pins giving information about the program progress:

- **gmoccapy.program.length** *(sint OUT)* - Shows the total number of lines of the program.
- **gmoccapy.program.current-line** *(sint OUT)* - Indicates the current working line of the program.
- **gmoccapy.program.progress** *(real OUT)* - Gives the program progress in percentage, by line number.
- **gmoccapy.program.time-total** *(real OUT)* - The whole program's estimated run time, in seconds.
- **gmoccapy.program.time-remaining** *(real OUT)* - What is left of it from the line being executed, in seconds.
- **gmoccapy.program.progress-time** *(real OUT)* - The program progress in percentage, by time rather

than by line.

Die Werte sind möglicherweise nicht sehr genau, wenn Sie mit Unterroutinen oder großen Remap-Prozeduren arbeiten. Auch Schleifen verursachen unterschiedliche Werte.

The three time pins come from the estimate the G-code parse makes: it models the machine's velocity and acceleration limits, the corners between moves, the feed modes, dwells and tool changes, at 100% feed, rapid and spindle override. Overrides, adaptive feed and M0/M1/M66 waits are not modelled, and the three pins are zero on a machine whose INI states no velocity limits. The progress bar shows the same estimate - the percentage by time, and the time so far beside the run it is heading for - and falls back to counting lines when there is none.

Werkzeugbezogene Pins

Werkzeugwechsel-Pins

Diese Pins werden bereitgestellt, um den GMOCCAPY-internen Werkzeugwechsel-Dialog zu verwenden, der dem von AXIS bekannten Dialog ähnelt, jedoch mit einigen Änderungen. So erhalten Sie nicht nur die Meldung, dass Sie zu *Werkzeug Nummer 3* wechseln sollen, sondern auch die Beschreibung dieses Werkzeugs wie *7,5 mm 3-Nuten-Fräser*. Die Informationen werden aus der Werkzeugtabelle entnommen, es liegt also an Ihnen, was angezeigt wird.

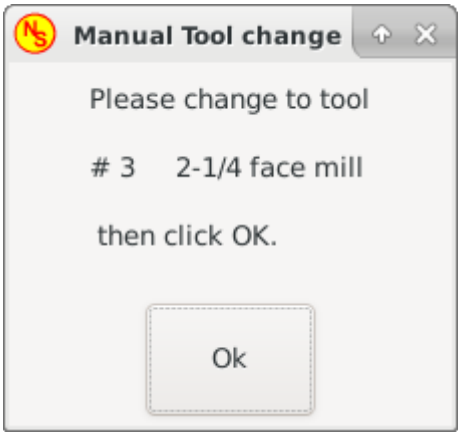


Figure 163. GMOCCAPY Werkzeugwechsel-Dialog

- **gmoccapy.toolchange-number** *(sint IN)* - The number of the tool to be changed
- **gmoccapy.toolchange-change** *(bool IN)* - Indicates that a tool has to be changed
- **gmoccapy.toolchange-changed** *(bool OUT)* - Indicates tool has been changed
- **gmoccapy.toolchange-confirm** *(bool IN)* - Confirms tool change

Normalerweise werden sie für einen manuellen Werkzeugwechsel so angeschlossen:

```
net tool-change gmoccapy.toolchange-change <= iocontrol.0.tool-change
net tool-changed gmoccapy.toolchange-changed <= iocontrol.0.tool-changed
net tool-prep-number gmoccapy.toolchange-number <= iocontrol.0.tool-prep-number
net tool-prep-loop iocontrol.0.tool-prepare <= iocontrol.0.tool-prepared
```

NOTE Bitte beachten Sie, dass diese Verbindungen in der postgui HAL Datei vorgenommen

werden müssen.

Werkzeug-Offset Pins

Mit diesen Pins können Sie die aktiven Werkzeugversatzwerte für X und Z im Werkzeuginformationsrahmen anzeigen. Sie sollten wissen, dass sie erst aktiv sind, nachdem G43 gesendet wurde.

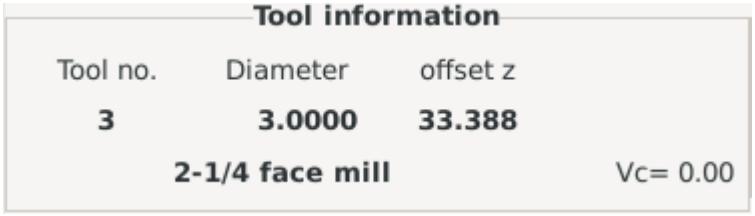


Figure 164. Informationsbereich für Werkzeuge

- **gmoccapy.tooloffset-x** *(real IN)*
- **gmoccapy.tooloffset-z** *(real IN)*

NOTE Die Zeile "tooloffset-x" wird bei einer Fräsmaschine nicht benötigt und wird bei einer Fräsmaschine mit trivialer Kinematik nicht angezeigt.

Um die aktuellen Offsets anzuzeigen, müssen die Pins in der postgui-HAL-Datei wie folgt verbunden werden:

```
net tooloffset-x gmoccapy.tooloffset-x <= motion.tooloffset.x
net tooloffset-z gmoccapy.tooloffset-z <= motion.tooloffset.z
```

IMPORTANT Bitte beachten Sie, dass GMOCCAPY selbst dafür sorgt, die Offsets zu aktualisieren, indem es nach jedem Werkzeugwechsel ein G43 sendet, **aber nicht im Auto-Modus!**
Wenn Sie also ein Programm schreiben, sind Sie dafür verantwortlich, nach jedem Werkzeugwechsel ein G43 zu senden!

10.2.6. Automatische Werkzeugvermessung

GMOCCAPY bietet eine integrierte automatische Werkzeugmessung. Um diese Funktion zu nutzen, müssen Sie einige zusätzliche Einstellungen vornehmen und vielleicht möchten Sie den angebotenen HAL-Pin verwenden, um Werte in Ihrem eigenen NGC-Remap-Verfahren zu erhalten.

IMPORTANT Vergessen Sie nicht, vor dem ersten Test die Tasterhöhe und die Tastergeschwindigkeiten auf der Einstellungsseite einzugeben! Siehe Einstellungsseite Werkzeugmessung.

Es könnte auch eine gute Idee sein, einen Blick auf das Video zur Werkzeugvermessung zu werfen, siehe Videos zur Werkzeugvermessung.

Die Werkzeugmessung in GMOCCAPY läuft ein wenig anders ab als in vielen anderen GUIs. Sie sollten

diese Schritte befolgen:

1. Berührung des Werkstücks in X und Y.
2. Messen Sie die Höhe Ihres Blocks von der Basis, an der sich Ihr Werkzeugschalter befindet, bis zur Oberseite des Blocks (einschließlich Spannfutter usw.).
3. Drücken Sie die Taste Blockhöhe und geben Sie den Messwert ein.
4. Gehen Sie in den Automatikmodus und starten Sie Ihr Programm.

Hier ist eine kleine Skizze:

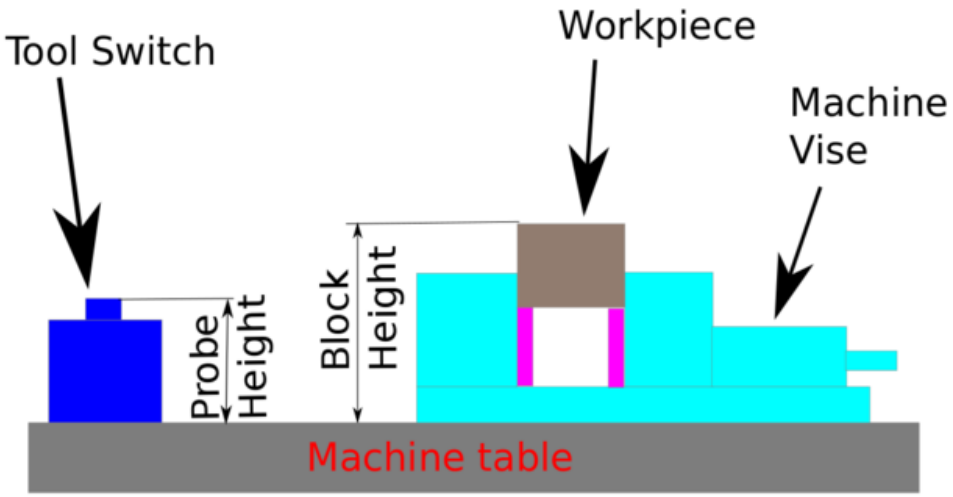


Figure 165. Werkzeugmessung Daten

Beim ersten vorgegebenen Werkzeugwechsel wird das Werkzeug vermessen und der Versatz automatisch auf die Blockhöhe eingestellt. Der Vorteil der GMOCCAPY Methode ist, dass Sie kein Referenzwerkzeug benötigen.

NOTE Ihr Programm muss am Anfang einen Werkzeugwechsel enthalten! Das Werkzeug wird vermessen, auch wenn es schon vorher benutzt wurde, so dass keine Gefahr besteht, wenn sich die Blockhöhe geändert hat. Es gibt mehrere Videos auf YouTube, die zeigen, wie man das macht.

Verfügbare Pins

GMOCCAPY bietet fünf Pins für die Werkzeugmessung. Diese Pins werden meist von einem G-Code-Unterprogramm gelesen, damit der Code auf verschiedene Werte reagieren kann.

- **gmoccapy.toolmeasurement** *(bool OUT)* - Enable or not tool measurement
- **gmoccapy.blockheight** *(real OUT)* - The measured value of the top face of the workpiece
- **gmoccapy.probeheight** *(real OUT)* - The probe switch height
- **gmoccapy.searchvel** *(real OUT)* - The velocity to search for the tool probe switch
- **gmoccapy.probevel** *(real OUT)* - The velocity to probe tool length

Änderungen an der INI-Datei

Ändern Sie Ihre INI-Datei so, dass sie die folgenden Abschnitte enthält.

Der RS274NGC-Abschnitt

```
[RS274NGC]
# Unterfunktion wird aufgerufen, wenn ein Fehler beim Werkzeugwechsel auftritt, wird
nicht bei jeder Maschinenkonfiguration benötigt
ON_ABORT_COMMAND=O <on_abort> call

# The remap code
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=change epilog=change_epilog
```

NOTE Stellen Sie sicher, dass INI_VARS und HAL_PIN_VARS nicht auf 0 gesetzt sind, sondern standardmäßig auf 1.

Der Abschnitt Werkzeugsensor (engl. tool sensor)

Die Position des Werkzeugsensors und die Startposition der Antastbewegung, alle Werte sind absolute Koordinaten, mit Ausnahme von MAXPROBE, welche in relativer Bewegung angegeben werden müssen.

```
[TOOLSENSOR]
X = 10
Y = 10
Z = -20
MAXPROBE = -20
```

Der Abschnitt "Position ändern"

Die Position wird nicht absichtlich TOOL_CHANGE_POSITION genannt - **canon verwendet diesen Namen und stört sonst.** Die Position, an welche die Maschine bewegt werden soll, bevor der Befehl zum Werkzeugwechsel gegeben wird. Alle Werte sind in absoluten Koordinaten.

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

Die Python-Sektion

Das Python-Plug-in dient als Interpreter und Task.

```
[PYTHON]
# The path to start a search for user modules
PATH_PREPEND = python
# The start point for all.
TOPLEVEL = python/toplevel.py
```

Benötigte Dateien

Erstellen Sie zunächst ein Verzeichnis "python" in Ihrem Konfigurationsverzeichnis. Kopieren Sie von

<Ihr_linuxcnc-dev_verzeichnis>/configs/sim/gmoccapy/python die folgenden Dateien in den gerade erstellten Ordner config_dir/python:

- toplevel.py
- remap.py
- stdglue.py

Von <Ihr_linuxcnc-dev_Verzeichnis>/configs/sim/gmoccapy/macros kopieren

- on_abort.ngc
- change.ngc

in das als SUBROUTINE_PATH angegebene Verzeichnis, siehe RS274NGC Section.

Öffnen Sie change.ngc mit einem Editor und entfernen Sie die Kommentare in den folgenden Zeilen (49 und 50):

```
F #<_hal[gmoccapy.probevel]>
G38.2 Z-4
```

Möglicherweise möchten Sie diese Datei an Ihre Bedürfnisse anpassen.

Benötigte HAL-Verbindungen

Schließen Sie die Werkzeugsonde in Ihrer HAL-Datei wie folgt an:

```
net probe motion.probe-input <= <Ihr_input_pin>
```

Die Zeile könnte so aussehen:

```
net probe  motion.probe-input <= parport.0.pin-15-in
```

Fügen Sie in Ihrer postgui.hal-Datei die folgenden Zeilen hinzu:

```
# Die nächsten Zeilen werden nur benötigt, wenn die Pins vorher angeschlossen waren.
unlinkp iocontrol.0.tool-change
unlinkp iocontrol.0.werkzeug-gewechselt
unlinkp iocontrol.0.werkzeug-vorbereitung-nummer
unlinkp iocontrol.0.werkzeug-vorbereitet

# Verknüpfung mit GMOCCAPY toolchange, damit Sie den Vorteil der Werkzeugbeschreibung im
Änderungsdialog nutzen können
net tool-change gmoccapy.toolchange-change <= iocontrol.0.tool-change
net tool-changed gmoccapy.toolchange-changed => iocontrol.0.tool-changed
net tool-prep-number gmoccapy.toolchange-number <= iocontrol.0.tool-prep-number
net tool-prep-loop iocontrol.0.tool-prepare <= iocontrol.0.tool-prepared
```

10.2.7. Die Einstellungsseite

Um die Seite zu öffnen, müssen Sie auf  klicken und einen Freischaltcode eingeben, der standardmäßig **123** lautet. Wenn Sie diesen Code ändern wollen, müssen Sie die versteckte Einstellungsdatei bearbeiten, siehe the display section für Details.

Die Seite ist in drei Hauptregisterkarten unterteilt:

Erscheinungsbild

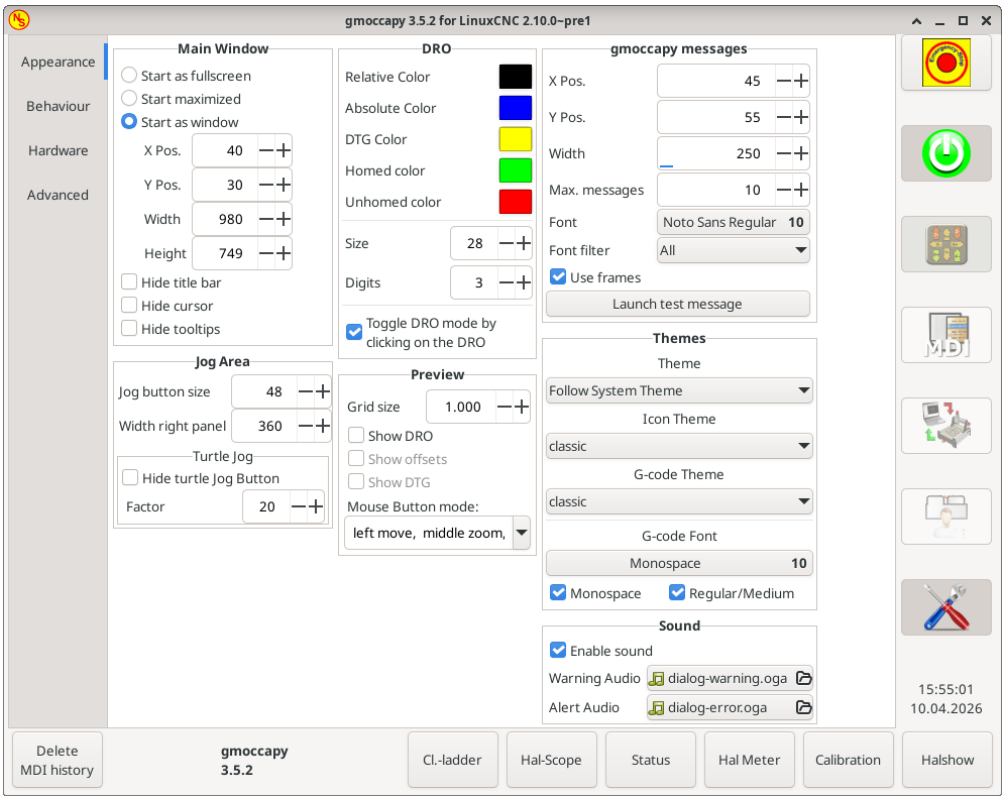


Figure 166. GMOCCAPY Einstellungsseite Erscheinungsbild

Auf dieser Registerkarte finden Sie die folgenden Optionen:

Hauptfenster

Hier können Sie auswählen, wie die grafische Benutzeroberfläche gestartet werden soll. Der Hauptgrund dafür war der Wunsch, eine einfache Möglichkeit für den Benutzer zu schaffen, die Startoptionen einzustellen, ohne dass er den Code berühren muss. Sie haben drei Optionen:

- *Start als Vollbildschirm*
- *Maximiert starten*
- *Start als Fenster* - Wenn Sie "Start als Fenster" wählen, werden die Spinboxen zum Einstellen der Position und Größe aktiv. Einmal eingestellt, startet die GUI jedes Mal an der gewählten Stelle und mit der gewählten Größe. Dennoch kann der Benutzer die Größe und Position mit der Maus verändern, was aber keinen Einfluss auf die Einstellungen hat.
- *hide title bar* - Erlaubt das Ausblenden der Titelleiste. (Voreinstellung: Titelleiste sichtbar).

- *Cursor ausblenden* - Blendet den Cursor aus, was sehr nützlich ist, wenn Sie einen Touchscreen verwenden.
- *hide tooltips* - Blendet die Tooltips aus.

Jog Area

- *Jog button size* - Change the size of the job buttons in the manual mode page
- *Width right panel* - Change the width of the right part of window (= width of the DRO)

Schildkröten Jog

Diese Einstellungen haben Einfluss auf die Jog-Geschwindigkeiten.

- *Schildkröten-Jog-Schaltfläche ausblenden* - Blendet die Schaltfläche rechts neben dem Schieberegler für die Jog-Geschwindigkeit aus. Wenn Sie diese Schaltfläche ausblenden, achten Sie bitte darauf, dass der „Hase-Modus“ aktiviert ist, da Sie sonst nicht schneller als die Schildkröte (engl. Turtle)-Jog-Geschwindigkeit (d.h. langsam) joggen können, die sich aus dem Turtle-Jog-Faktor ergibt.
- *Turtle-Jog-Faktor* - Legt den Maßstab für den Schildkröten-Jogging-Modus fest. Wenn Sie einen Faktor von 20 einstellen, beträgt die maximale Jog-Geschwindigkeit der Schildkröte 1/20 der maximalen Geschwindigkeit der Maschine im Schildkrötenmodus (Taste gedrückt, zeigt die Schildkröte).

NOTE Dieser Button kann über den turtle-jog HAL-Pin gesteuert werden.

DRO

Sie haben die Möglichkeit, die Hintergrundfarben für die verschiedenen DRO-Zustände auszuwählen. So können Benutzer, die an Protanopie (Rot/Grün-Schwäche) leiden, die richtigen Farben auswählen.

Voreingestellte Hintergrundfarben sind:

- *Relative Farbe* = black (engl. für schwarz)
- *Absolute Farbe* = blue (engl. für blau)
- *DTG Farbe* = yellow (engl. für gelb)

Die Vordergrundfarbe der DRO kann ausgewählt werden mit:

- *Homed Color* (engl. für Anzeigefarbe für eine erfolge Referenzierung) = green (engl. für grün)
- unhomed color (Anzeigefarbe für ausgebliebene Referenzierung) = red (engl. für rot)

NOTE Sie können durch die DRO-Modi (absolut, relativ, Entfernung zu gehen) wechseln, indem Sie auf die Zahl auf dem DRO klicken! Wenn Sie auf den linken Seitenbuchstaben des DRO klicken, können Sie in einem Popup-Fenster den Wert der Achsen festlegen, was das Einstellen des Werts erleichtert, da Sie nicht über die untere Touch-Off-Taste gehen müssen.

- Ermöglicht die Einstellung der Größe der DRO-Schrift, Standard ist 28, wenn Sie einen größeren Bildschirm verwenden, können Sie die Größe auf 56 erhöhen. Wenn Sie 4 Achsen verwenden, wird die DRO-Schriftgröße aus Platzgründen 3/4 des Wertes betragen.

- *digits* - Stellt die Anzahl der Ziffern der Positionsanzeige von 1 bis 5 ein.

NOTE	Imperiale Einheiten zeigen eine Ziffer mehr an als metrische. Wenn Sie also imperiale Maschineneinheiten verwenden und den Ziffernwert auf 1 setzen, erhalten Sie im metrischen System überhaupt keine Ziffer.

- *Toggle DRO mode by clicking on the DRO* - If not active, a mouse click on the DRO will not take any action.
  By default this checkbox is active, so every click on any DRO will toggle the DRO readout from actual to relative to DTG (distance to go).
  Neverthereless a click on the axis letter will open the popup dialog to set the axis value.

Vorschau

- *Grid Size* - Legt die Rastergröße des Vorschaufensters fest. Leider muss die Größe **in Zoll** eingestellt werden, auch wenn Ihre Maschine metrische Einheiten hat. Wir hoffen, dies in einer zukünftigen Version zu beheben.

NOTE	Das Gitter wird in der perspektivischen Ansicht nicht angezeigt.

- *Show DRO* - Zeigt die DRO auch im Vorschaufenster an, sie wird immer in der Vollbildvorschau angezeigt.
- *Show DTG* - Zeigt die DTG (direkte Entfernung zum Endpunkt) in der Vorschau, wenn Show DRO aktiv ist: Ansonsten nur Vorschau in voller Größe.
- *Offsets anzeigen* - Zeigt die Offsets im Vorschaubereich an, wenn Show DRO aktiv ist. Sonst Vorschau nur in voller Größe.
- *Mouse Button Mode* - Mit dieser Combobox können Sie das Schaltflächenverhalten der Maus zum Drehen, Verschieben oder Zoomen innerhalb der Vorschau auswählen:
  - Links drehen, Mitte verschieben, rechts zoomen
  - Links zoomen, Mitte verschieben, rechts rotieren
  - links bewegen, mitte drehen, rechts zoomen
  - Links zoomen, Mitte drehen, rechts bewegen
  - Links verschieben, Mitte zoomen, rechts rotieren
  - Links drehen, Mitte zoomen, rechts bewegen

Standardeinstellung ist links bewegen, Mitte zoom, rechts rotieren.
Mit dem Mausrad können Sie die Vorschau in jedem Modus vergrößern.

TIP	Wenn Sie ein Element in der Vorschau auswählen, wird das ausgewählte Element als Rotationsmittelpunkt genommen und im Automodus wird die entsprechende Codezeile hervorgehoben.

Gmoccapy Meldungen (engl. messages)

Daraufhin werden kleine Pop-up-Fenster mit einer Meldung oder einem Fehlertext angezeigt, ähnlich

denen, die von AXIS bekannt sind. Sie können eine bestimmte Meldung löschen, indem Sie auf die Schaltfläche "Schließen" klicken. Wenn Sie die letzte Meldung löschen möchten, drücken Sie einfach die Taste "Fenster" auf Ihrer Tastatur, oder löschen Sie alle Meldungen auf einmal mit "Strg + Leertaste".

Sie haben die Möglichkeit, einige Optionen einzustellen:

- *X Pos* - Die Position der oberen linken Ecke der Nachricht in X gezählt in Pixel von der oberen linken Ecke des Bildschirms.
- *Y Pos* - Die Position der oberen linken Ecke der Nachricht in Y gezählt in Pixel von der oberen linken Ecke des Bildschirms.
- *Width* - Die Breite des Meldungsfelds.
- *Max Messages* - Die maximale Anzahl von Nachrichten, die Sie gleichzeitig sehen wollen. Wenn Sie hier 10 eingeben, wird die 11. Nachricht die erste löschen, so dass Sie nur die letzten 10 sehen werden.
- Schriftart (engl. font) - Die Schriftart und -größe, die Sie für die Anzeige der Meldungen verwenden möchten.
- *Font filter* - A filter for the font chooser above.
- *Rahmen verwenden* (engl. use frames) - Wenn Sie das Kontrollkästchen aktivieren, wird jede Nachricht in einem Rahmen angezeigt, so dass es viel einfacher ist, die Nachrichten zu unterscheiden. Allerdings benötigen Sie dann etwas mehr Platz.
- *Launch test message*-button - Es zeigt eine Nachricht, so dass Sie die Änderungen Ihrer Einstellungen sehen können, ohne tatsächlich einen Fehler zu erzeugen.

Themen

- *Theme* - This lets the user select what desktop theme to apply. By default "Follow System Theme" is set.
- *Icon Theme* - Change the icon theme. Currently there are three themes available:
  - classic
  - Material
  - Material Light

  For more information and how to create custom icon themes, see section Icon Theme.
- *G-code Theme* - Select a colour theme for the G-code viewer/editor.
- *G-code Font* - Select a font for the G-code viewer/editor. Default is "monospace 10".

Sound

Select what error and messages sounds should be played and if sound should be played at all.

Behaviour

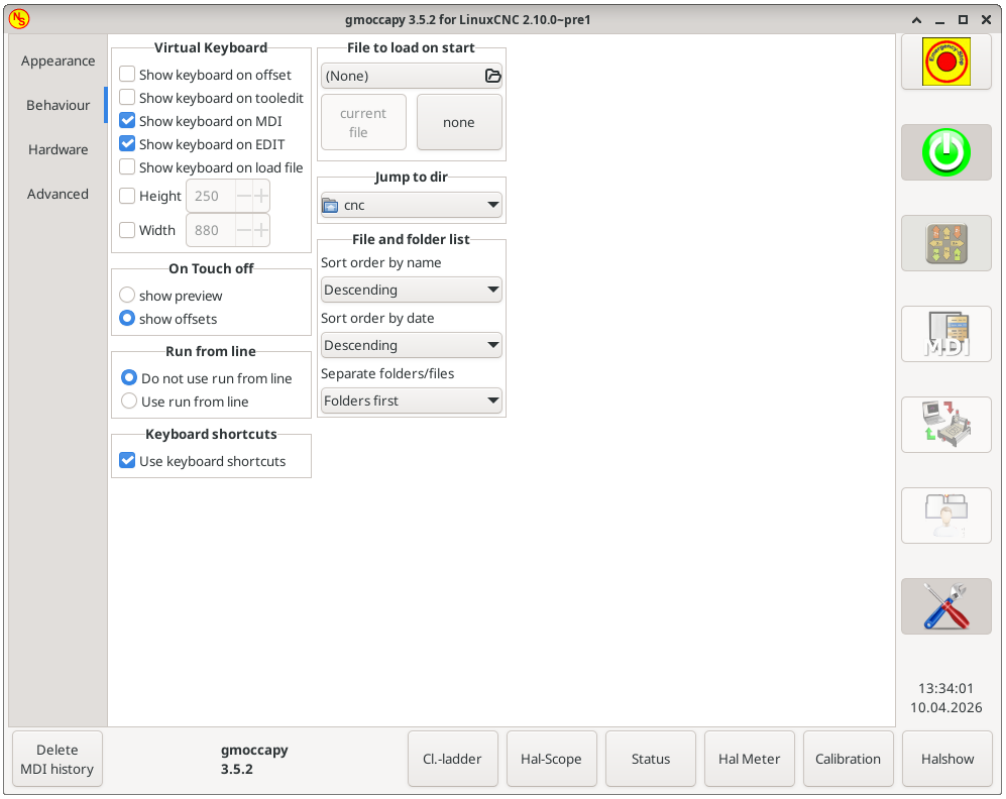


Figure 167. GMOCCAPY settings page Behaviour

Virtuelle Tastatur

Mit den Checkboxen kann der Benutzer auswählen, ob die integrierte Tastatur sofort angezeigt werden soll, wenn der MDI-Modus aufgerufen wird, die Offset-Seite aufgerufen wird, das Tooledit-Widget aufgerufen wird oder wenn ein Programm im EDIT-Modus geöffnet wird. Die Tastaturschaltfläche in der unteren Schaltflächenliste wird von diesen Einstellungen nicht beeinflusst, sodass Sie die Tastatur durch Drücken der Schaltfläche ein- oder ausblenden können.

Die Vorsteinstellung ist:

- *Zeige Tastatur bei Offsets* = False
- *Zeige Tastatur in Werkzeugverwaltung* = False
- *Zeige Tastatur bei MDI* = True
- *Zeige Tastatur beim Editieren* = True
- *Tastatur beim Laden der Datei anzeigen* = False

NOTE Wenn dieser Abschnitt nicht empfindlich ist, haben Sie keine virtuelle Tastatur installiert, unterstützt werden *onboard* und *matchbox-keyboard*.

NOTE Wenn das Tastaturlayout nicht korrekt ist, d.h. ein Klick auf Y ergibt Z, dann wurde das Layout nicht richtig eingestellt, was mit Ihren Gebietsschemaeinstellungen zusammenhängt. Für Onboard kann dies mit einer kleinen Batch-Datei mit folgendem Inhalt gelöst werden:

```
#!/bin/bash
```

```
setxkbmap -model pc105 -layout de -variant basic
```

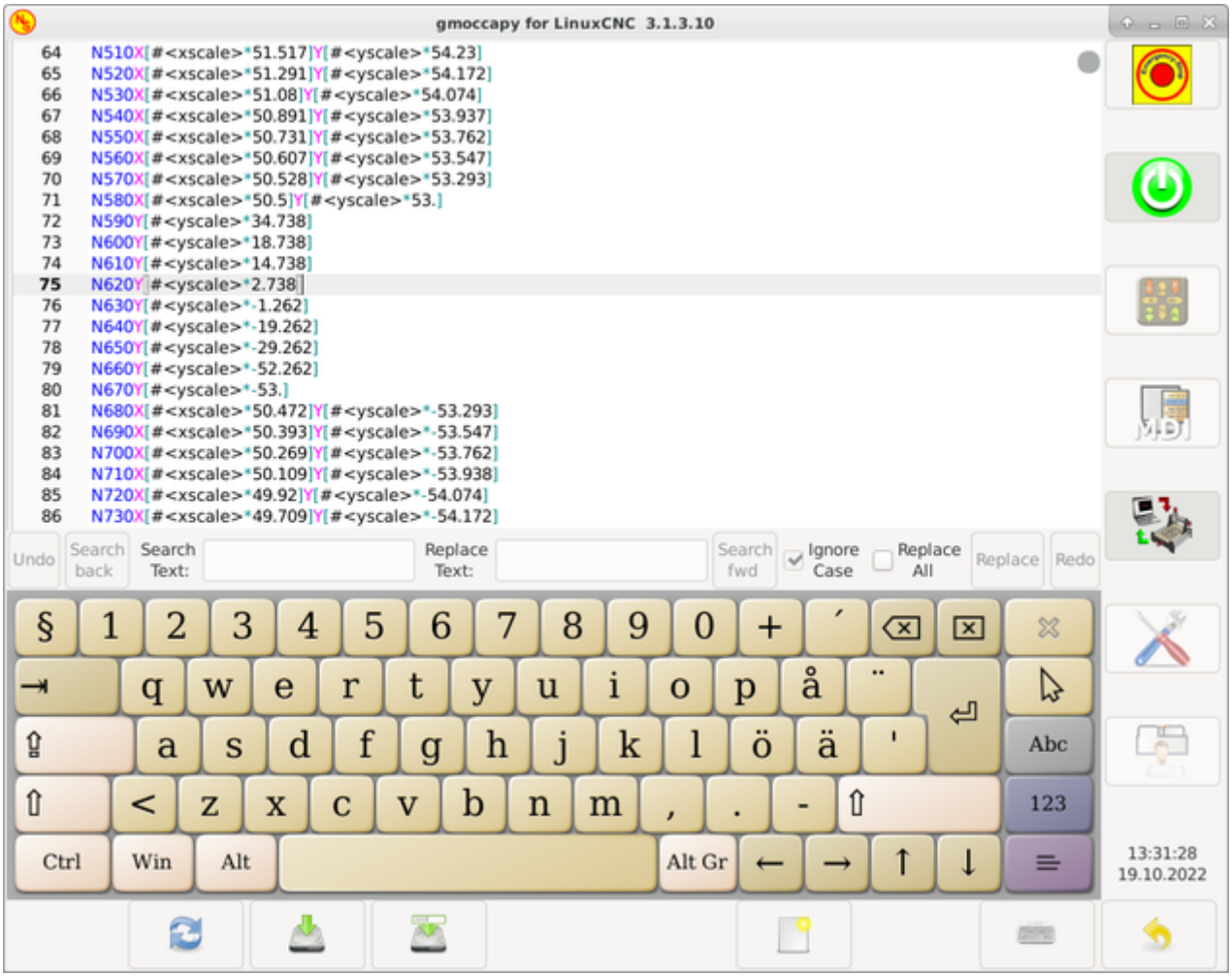
Die Buchstaben "de" sind für Deutsch, Sie müssen sie entsprechend Ihrer Locale-Einstellungen setzen. Führen Sie einfach diese Datei vor dem Start von LinuxCNC, kann es auch das Hinzufügen eines Starters zu Ihrem lokalen Ordner getan werden.

```
./config/autostart
```

Damit das Layout beim Start automatisch eingestellt wird.

Für matchbox-keyboard müssen Sie Ihr eigenes Layout erstellen, für ein deutsches Layout fragen Sie im Forum.

GMOCCAPY mit Onboard-Tastatur im Bearbeitungsmodus



On Touch Off

Hier können Sie wählen, ob die Registerkarte "Vorschau" oder die Registerkarte "Offset-Seite" angezeigt werden soll, wenn Sie wenn Sie durch Klicken auf die entsprechende untere Schaltfläche in den Touch-off-Modus wechseln.

- *Vorschau anzeigen*
- *Offsets anzeigen*

Ausführen von gegebener Zeile

Sie können die Ausführung aus der Zeile erlauben oder verbieten. Dadurch wird die entsprechende Schaltfläche unempfindlich (ausgegraut), so dass der Benutzer diese Option nicht verwenden kann. Die Standardeinstellung ist "Aus der Zeile ausführen" deaktivieren.

WARNING	Es ist nicht empfehlenswert, "Ab Zeile ausführen" (engl. run from line) zu verwenden, da LinuxCNC sich nicht um alle vorherigen Zeilen im Code vor der Startzeile kümmert. Daher sind Fehler oder Abstürze recht wahrscheinlich.

Tastatürkürzel

Einige Benutzer möchten ihre Maschine mit den Tasten der Tastatur steuern, andere werden dies niemals zulassen. Jeder kann also wählen, ob er sie verwenden möchte oder nicht.
Tastaturkürzel sind standardmäßig deaktiviert. Sie sind zu aktivieren durch die checkbox

- *Nutzen von Tastaturkürzeln* (engl. *Use keyboard shortcuts*)

WARNING	Es wird nicht empfohlen, Tastatur-Jogging zu verwenden, da dies ein ernsthaftes Risiko für Bediener und Maschine darstellt.

Bitte achten Sie darauf, wenn Sie eine Drehmaschine benutzen, dann sind die Abkürzungen anders. Siehe das Kapitel zu Drehmaschinen.

Allgemeines

- *F1* - Löst Notaus aus (engl. E-stop) (funktioniert auch, wenn die Tastaturkürzel deaktiviert sind)
- *F2* - Maschine ein-/ausschalten
- *F3* - manueller Modus
- *F5* - MDI-Modus
- *ESC* - Abbruch (engl. abort)

Im manuellen Modus

- *Pfeil nach links* oder *NumPad_Left* - Jog X minus
- *Pfeil rechts* oder *NumPad_rechts* - Jog X plus
- *Arrow_up* oder *NumPad_Up* - Jog Y plus
- *Pfeil nach unten* oder *NumPad_nach unten* - Jog Y minus
- *Page_Up* oder *NumPad_Page_Up* - Jog Z plus
- *Page_Down* oder *NumPad_Page_Down* - Jog Z minus

Im Auto-Modus

- *R* oder *r* - Programm starten
- *P* oder *p* - Programm anhalten
- *S* oder *s* - Programm fortsetzen

- *Control* + *R* oder *Control* + *r* - Datei erneut laden

Nachrichtenverarbeitung (siehe Nachrichtenverhalten und -darstellung)

- *WINDOWS* - Letzte Nachricht löschen
- *Control* + *Space* - Alle Nachrichten löschen

Beim Starten zu ladende Datei

Wählen Sie die Datei, die beim Starten geladen werden soll. Wenn eine Datei geladen ist, kann sie durch Drücken der aktuellen Taste eingestellt werden. Um zu verhindern, dass ein Programm beim Start geladen wird, drücken Sie einfach die Taste None.

The file selection screen will use the filters you have set in the INI file, if there aren't any filters given, you will only see **NGC files**. The path will be set according to the INI settings in `[DISPLAY] PROGRAM_PREFIX`.
NGC files can be explicitly excluded from the file selection screen by passing **no_ngc** instead of an extension:

```
PROGRAM_EXTENSION = no_ngc Exclude ngc files from selection screen
```

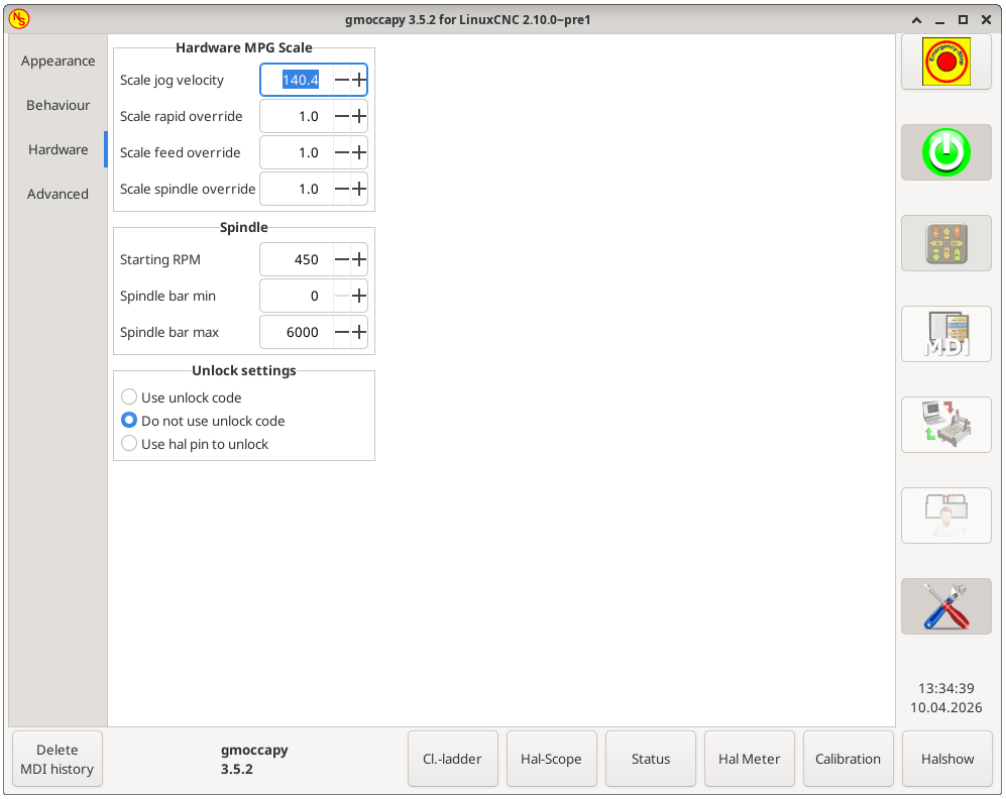
Zum Verzeichnis springen

Sie können hier das Verzeichnis einstellen, in das gesprungen werden soll, wenn Sie im Dateiauswahldialog auf die entsprechende Schaltfläche klicken.

File and folder list

For the file chooser page it is possible to change the sorting options with these parameters.

Hardware



Hardware MPG-Skalierung

Für die verschiedenen HAL-Pins, an die MPG Wheels angeschlossen werden, können Sie einzelne Skalen auswählen, die angewendet werden sollen. Der Hauptgrund dafür war mein eigener Versuch, dies durch HAL-Verbindungen zu lösen, was zu einer sehr komplexen HAL-Datei führte. Stellen Sie sich vor, ein Benutzer hat ein MPG-Rad mit 100 PPR (Pulse pro Rotation) und möchte die Höchstgeschwindigkeit von 14000 auf 2000 mm/min verlangsamen, das erfordert 12000 Pulse, was 120 Umdrehungen des Rades ergibt! Oder ein anderer Benutzer hat ein MPG-Rad mit 500 IPR und möchte die Spindelübersteuerung einstellen, die Grenzwerte von 50 bis 120 % hat, so dass er innerhalb von 70 Pulsen von der Minimal- zur Maximalgeschwindigkeit gelangt, was nicht einmal eine 1/4 Umdrehung bedeutet.

Standardmäßig werden alle Skalen anhand der Berechnung festgelegt:

```
(MAX - MIN)/100
```

Spindel

- Die *Startdrehzahl* (engl. starting RPM) - Legt die Drehzahl fest, die verwendet wird, wenn die Spindel gestartet wird und kein S-Wert eingestellt wurde.

> **NOTE** Dieser Wert wird entsprechend Ihrer Einstellungen in `[DISPLAY] DEFAULT_SPINDLE_SPEED` in Ihrer INI-Datei voreingestellt. Wenn Sie die Einstellungen auf der Einstellungsseite ändern, wird dieser Wert von diesem Moment an voreingestellt, Ihre INI-Datei wird nicht geändert.

- *Spindelstange min* und *Spindelstange max* - Legt die Grenzen der Spindelstange fest, die im INFO-Rahmen auf dem Hauptbildschirm angezeigt werden.

Standardwerte sind
MIN = 0
MAX = 6000

NOTE Es ist kein Fehler, falsche Werte anzugeben. Wenn Sie einen Maximalwert von 2000 eingeben und Ihre Spindel 4000 Umdrehungen pro Minute macht, wird nur der Stangenpegel bei höheren Drehzahlen als 2000 U/min falsch sein.

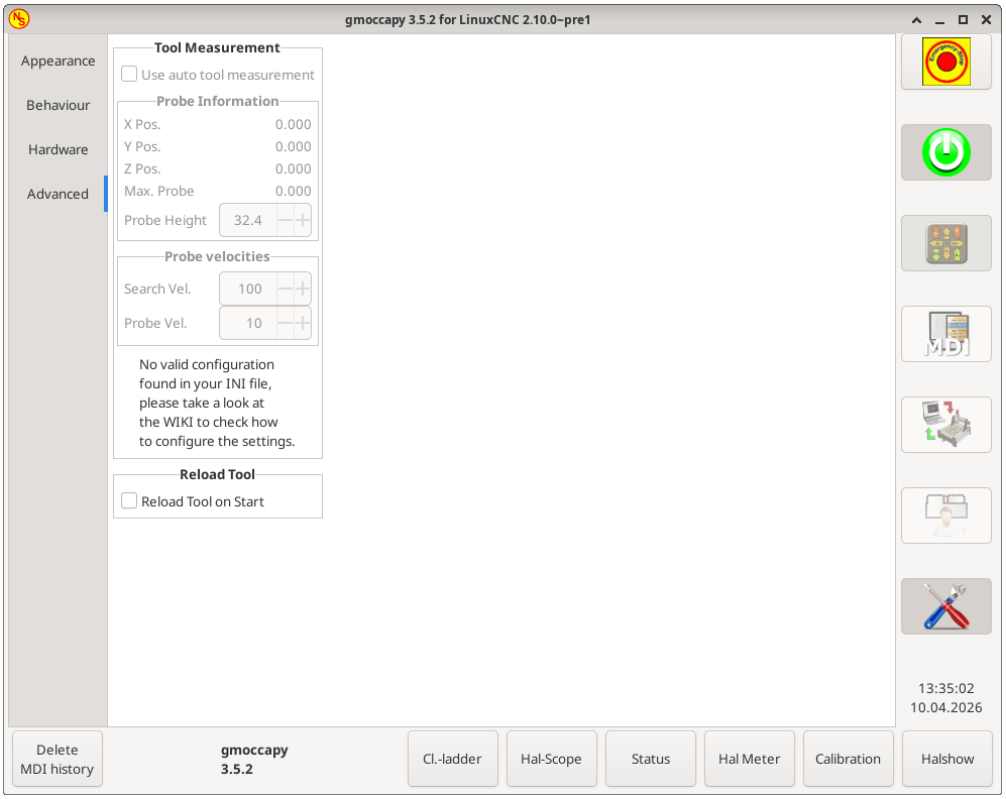
Unlock settings

Es gibt drei Optionen zum Entsperren der Einstellungsseite:

- *Freischaltcode verwenden* - der Benutzer muss einen Code eingeben, um Zugang zu erhalten.
- *Verwenden Sie keinen Freischaltcode* - Es findet keine Sicherheitsüberprüfung statt.
- *HAL-Pin zum Entsperren verwenden* - Hardware-Pin muss high sein, um die Einstellungen zu entsperren, siehe hardware unlock pin.

Standard ist, den *Entsperrcode zu verwenden* (Standard Code ist **123**).

Erweiterte Einstellungen



Werkzeugmessung

Bitte prüfen Sie Auto Tool Measurement

NOTE Wenn dieser Teil nicht empfindlich ist, haben Sie keine gültige INI-Datei-Konfiguration für die Verwendung der Werkzeugmessung.

- *Automatische Werkzeugvermessung verwenden* - Wenn diese Option aktiviert ist, wird nach jedem Werkzeugwechsel eine Werkzeugvermessung durchgeführt. Das Ergebnis wird in der Werkzeugtabelle gespeichert und nach dem Wechsel wird ein G43 ausgeführt.

Sonden-Informationen

Die folgenden Informationen werden aus Ihrer INI-Datei entnommen und müssen in absoluten Koordinaten angegeben werden:

- *X Pos.* - Die X-Position des Werkzeugschalters.
- *Y Pos.* - Die Y-Position des Werkzeugschalters.
- *Z Pos.* - Die Z-Position des Werkzeugschalters, wir fahren im Eilgang zu dieser Koordinate.
- *Max. Probe* - Die Entfernung, in der nach einem Kontakt gesucht werden soll; wird kein Kontakt in diesem Bereich gegeben ist, so wird ein Fehler ausgelöst. Die Entfernung muss in relativen Koordinaten angegeben werden, wobei die Bewegung von der Z-Position aus beginnt, d.h. Sie müssen einen negativen Wert angeben, um nach unten zu gehen!
- Sondenhöhe_ - Die Höhe Ihres Sondenschalters, Sie können sie messen. Tippen Sie einfach die Basis an, auf der sich der Tasterschalter befindet, und setzen Sie den Wert auf Null. Führen Sie dann einen Werkzeugwechsel durch und beobachten Sie den Wert tool_offset_z, das ist die Höhe, die Sie hier eingeben müssen.

Geschwindigkeiten der Sonde

- *Such-Geschwindigkeit* (engl. search velocity) - Die Geschwindigkeit für die Suche nach dem Werkzeugschalter, nach dem Kontakt wird das Werkzeug wieder nach oben gehen und dann geht in Richtung der Sonde wieder mit Sonden-Geschwindigkeit, so erhalten Sie bessere Ergebnisse.
- *Sonden-Geschw.* (engl. probe vel.) - Ist die Geschwindigkeit für die zweite Bewegung zum Schalter. Sie sollte langsamer sein, um bessere Berührungsergebnisse zu erzielen. Im Simulationsmodus ist dies in macros/change.ngc auskommentiert, da der Benutzer sonst zweimal auf den Taster klicken müsste.

Reload Tool (Werkzeug nochmal laden)::

- *Reload Tool on Start* - Lädt das letzte Werkzeug beim Start nach der Referenzfahrt.

Wenn diese Option aktiviert ist, wird das Werkzeug in der Spindel bei jeder Änderung der Voreinstellungsdatei gespeichert, so dass das zuletzt montierte Werkzeug beim Start wieder geladen werden kann. Das Werkzeug wird geladen, nachdem alle Achsen referenziert wurden, da es vorher nicht erlaubt ist, MDI-Befehle auszuführen. Wenn Sie NO_FORCE_HOMING verwenden, können Sie diese Funktion nicht nutzen, da das benötigte all_homed_signal nie ausgegeben wird.

10.2.8. Icon Themen

Icon-Themen werden verwendet, um das Aussehen der Icons von GMOCCAPY anzupassen.

GMOCCAPY wird mit drei verschiedenen Symbolthemen geliefert:

- *classic* - Die klassischen GMOCCAPY-Symbole.

- *material* - Ein modernes Icon-Thema, das von Googles Material Icons inspiriert ist und seine Farbgebung automatisch an das ausgewählte Desktop-Thema anpasst.
- *material-light* - Abgeleitet von material, aber optimiert für helle Desktop-Themen.

Das in GMOCCAPY verwendete Icon-Theme ist ein reguläres GTK Icon-Theme, das der Spezifikation des freedestktop Icon-Theme folgt. Somit kann jedes gültige GTK-Icon-Theme als GMOCCAPY-Icon-Theme verwendet werden, solange es die erforderlichen Icons enthält.

GMOCCAPY durchsucht die folgenden Verzeichnisse nach Icon-Themes:

- linuxcnc/share/gmoccapy/icons
- ~/.icons

Benutzerdefiniertes Symboldesign (Icon Theme)

Das Erstellen eines eigenen Icon-Theme ist ziemlich einfach. Alles, was Sie brauchen, ist ein Texteditor und natürlich die gewünschten Icons als Pixel- oder Vektorgrafiken. Details darüber, wie genau ein Icon-Theme aufgebaut ist, finden Sie unter Freedesktop Icon Theme Specification.

Beginnen Sie damit, ein leeres Verzeichnis mit dem Namen des Icon-Theme zu erstellen. Legen Sie das Verzeichnis in eines der Icon-Theme-Verzeichnisse von GMOCCAPY. Dann brauchen wir eine Datei namens index.theme im Stammverzeichnis unseres Icon-Themes mit den erforderlichen Metadaten für das Theme. Das ist eine einfache Textdatei mit mindestens den folgenden Abschnitten:

- [Icon Theme]

```
[Icon Theme]
Name=Ihr_Theme-Name
Comment=Eine Beschreibung des Themes
Inherits=hicolor
Directories=16x16/actions,24x24/actions,32x32/actions,48x48/actions,scalable/actions
```

  - Name: Der Name Ihres Icon-Designs.
  - Comment: Eine Beschreibung des Themas Ihres Symbols.
  - Inherits: Ein Icon-Thema kann von einem anderen Icon-Thema abgeleitet werden, der Standard ist hicolor.
  - Verzeichnisse: Eine durch Kommata getrennte Liste aller Verzeichnisse Ihres Icon-Themas.
    Jedes Verzeichnis enthält in der Regel alle Icons des Themas in einer bestimmten Größe, z.B. 16x16/actions sollte alle Icons mit der Kategorie "actions" in der Größe 16x16 Pixel als Pixelgrafiken (z.B. png-Dateien) enthalten. Ein Sonderfall ist das Verzeichnis "scalable/actions", dieses enthält skalierbare Icons, die nicht an eine bestimmte Größe gebunden sind (z.B. svg-Dateien).
    Durch die Bereitstellung unterschiedlich großer Versionen der Icons können wir ein schön aussehendes Icon in verschiedenen Größen garantieren, und wir haben auch die Möglichkeit, das Icon entsprechend seiner Größe zu verändern, zum Beispiel kann ein 64x64 px großes Icon mehr Details enthalten als seine 16x16 px Version.

- Für jedes Verzeichnis müssen wir auch einen Abschnitt in die Datei index.theme schreiben:

```
[16x16/actions]
Size=16
Type=Fixed
Context=Actions

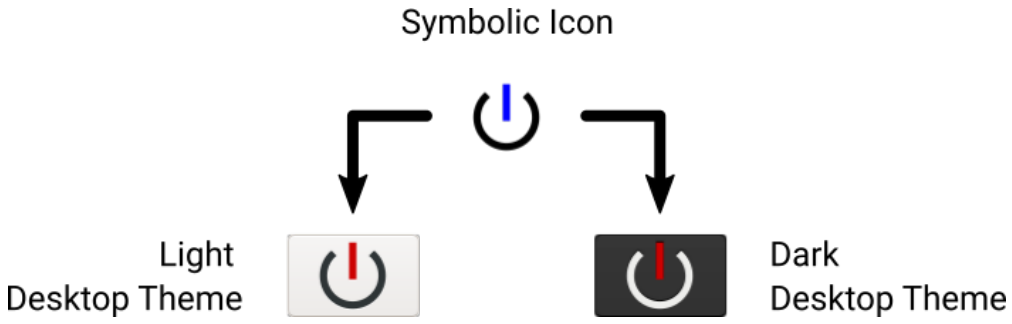
[scalable/actions]
Size=48
Type=Scalable
Context=Actions
```

  - SIze: Nominale Symbolgröße in diesem Verzeichnis
  - Type: Fixed (engl. für festgelegt), Threshold (engl. für Schwellwert) or Scalable (engl. für skalierbar)
  - Context: Beabsichtigte "Kategorie" von Icons

Das ist im Grunde alles, was man braucht, um ein eigenes Icon Theme zu erstellen.

Symbolische Icons

Symbolische Symbole sind eine besondere Art von Symbolen, in der Regel ein einfarbiges Bild. Das Besondere an symbolischen Icons ist, dass die Icons zur Laufzeit automatisch eingefärbt werden, um dem Desktop-Thema zu entsprechen. Auf diese Weise können Icon-Themen erstellt werden, die sowohl mit dunklen als auch mit hellen Desktop-Themen gut funktionieren (tatsächlich ist das nicht immer die beste Option, deshalb gibt es ein spezielles "Material-light"-Thema).



Um die symbolische Funktion zu nutzen, muss eine Symboldatei die Endung .symbolic.$ext haben (wobei $ext die reguläre Dateierweiterung wie png ist), zum Beispiel "power_on.symbolic.png".

Mit diesem Namen behandelt GTK dieses Bild als symbolisches Icon und wendet beim Laden des Icons eine Umfärbung an. Es gibt nur vier Farben, die verwendet werden dürfen:

Farbe	Hexadezimale RGB-Anteile	Beschreibung
black (engl. für schwarz)	`#000000`	Primäre Farbe, wird an die Primärfarbe des Desktop-Themas angepasst.
red (engl. für rot)	`#ff0000`	Erfolg: diese Farbe zeigt "Erfolg" an (normalerweise etwas Grünes).

Farbe	Hexadezimale RGB-Anteile	Beschreibung
green (engl. für grün)	`#00ff00`	Warnung: diese Farbe bedeutet "Warnung" (normalerweise etwas Gelbes/Oranges).
blue (engl. für blau)	`#0000ff`	Fehler: diese Farbe zeigt "Fehler" an (normalerweise etwas Rotes).

TIP Beispiele für symbolische Icons finden Sie unter `linuxcnc/share/gmoccapy/icons/material`.

10.2.9. Drehmaschinen-spezifischer Abschnitt

Wenn in der INI-Datei `LATHE = 1` angegeben wird, ändert die GUI ihr Aussehen entsprechend den speziellen Anforderungen einer Drehmaschine. Vor allem die Y-Achse wird ausgeblendet und die Jog-Buttons werden in einer anderen Reihenfolge angeordnet.

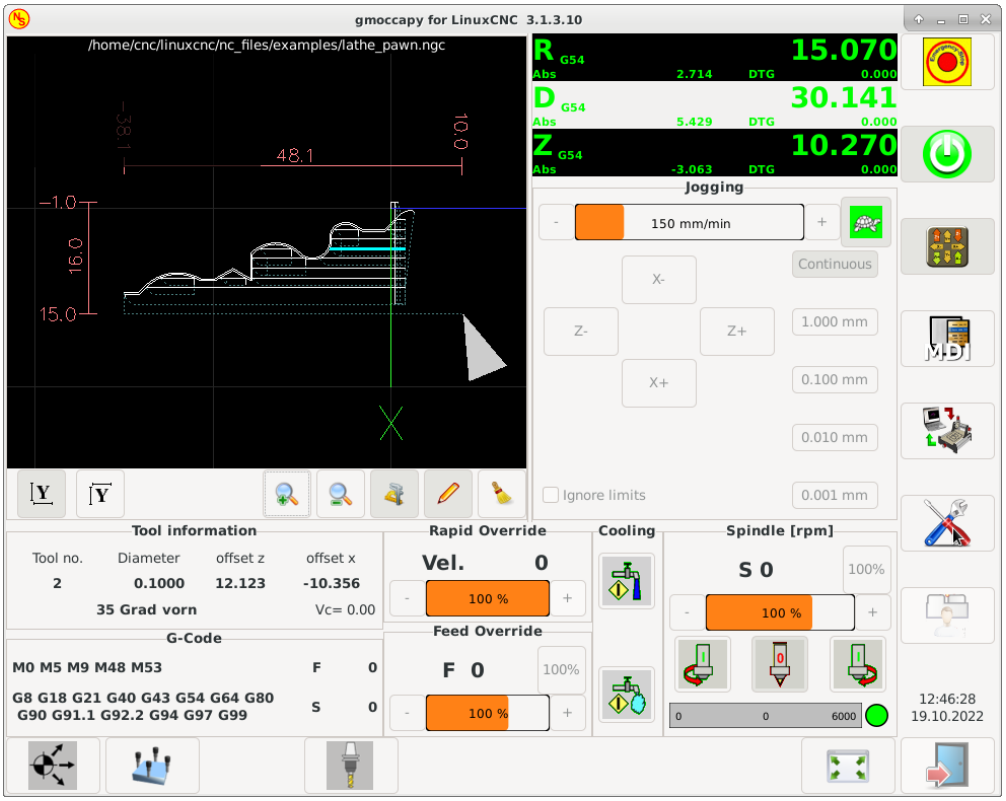


Figure 168. Normale Drehmaschine

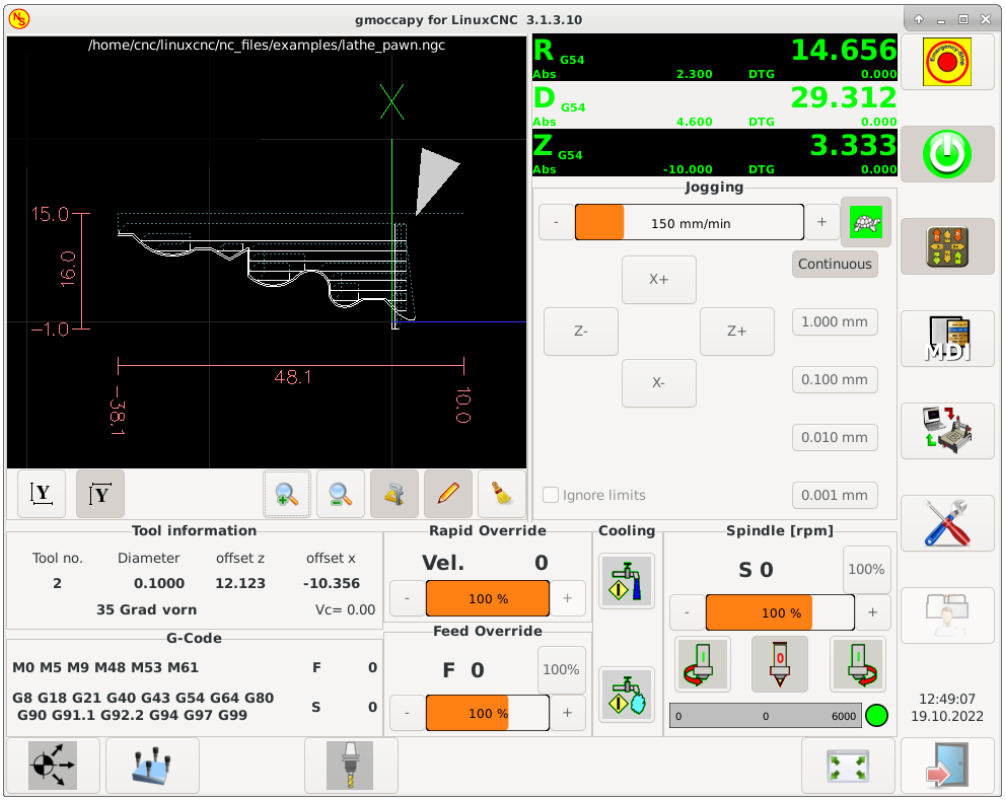


Figure 169. Drehmaschine mit hinterem Werkzeug

Wie Sie sehen, hat die R DRO einen schwarzen Hintergrund und die D DRO ist grau. Dies ändert sich je nach dem aktiven G-Code G7 oder G8. Der aktive Modus ist durch den schwarzen Hintergrund sichtbar, d.h. in den gezeigten Bildern ist G8 aktiv.

Der nächste Unterschied zum Standardbildschirm ist die Position des Jog-Buttons. X und Z haben ihre Plätze getauscht und Y ist verschwunden. Sie werden feststellen, dass die Schaltflächen X+ und X- ihren Platz je nach normaler oder hinterer Werkzeugdrehbank wechseln.

Auch das Verhalten der Tastatur wird sich ändern:

Normale Drehmaschine:

- *Pfeil links* oder *NumPad_links* - Jog Z minus
- *Pfeil rechts* oder *NumPad_rechts* - Jog Z plus
- *Pfeil nach oben* oder *NumPad_Hoch* - Jog X minus
- *Pfeil nach unten* oder *NumPad_runter - Jog X plus*

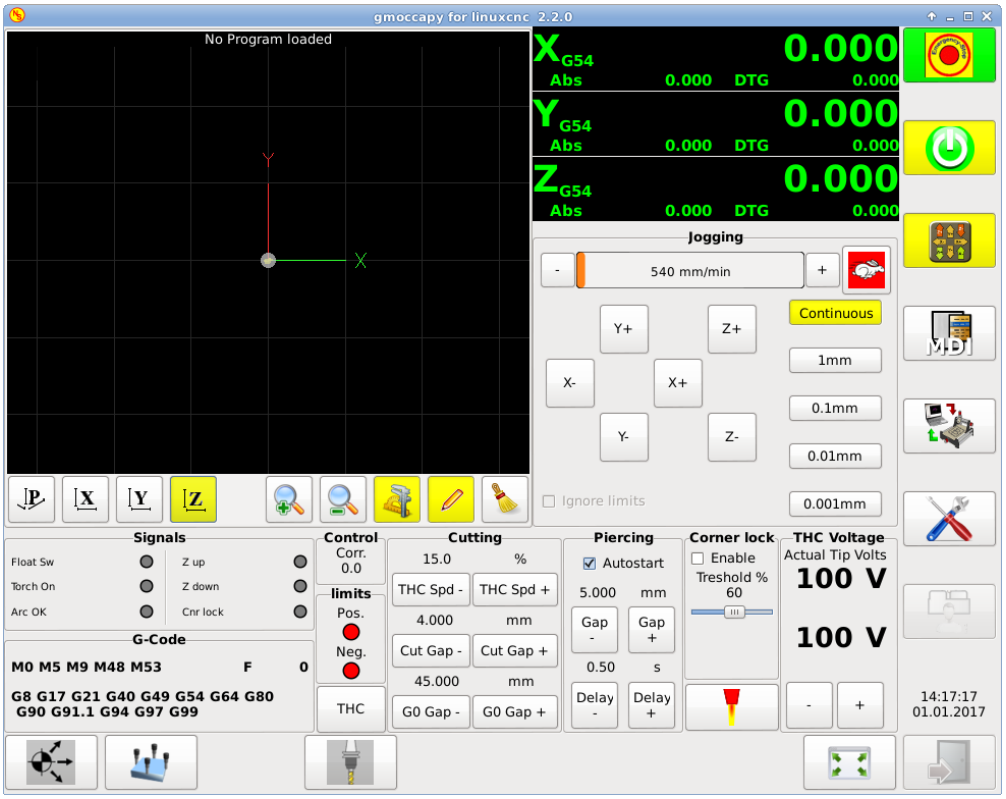
Drehmaschine mit hinterem Werkzeug:

- *Pfeil links* oder *NumPad_links* - Jog Z minus
- *Pfeil rechts* oder *NumPad_rechts* - Jog Z plus
- *Pfeil nach oben* oder *NumPad_Up* - Jog X plus
- *Pfeil nach unten* oder *NumPad_Down* - Jog X minus

Der Werkzeuginformationsrahmen zeigt nicht nur die Z-Korrektur, sondern auch die X-Korrektur und

die Werkzeugtabelle zeigt alle drehbankrelevanten Informationen an.

10.2.10. Plasmaspezifischer Abschnitt



Es gibt ein sehr gutes WIKI, das von Marius gepflegt wird, siehe Plasma wiki page.

10.2.11. Videos auf YouTube

Nachstehend finden Sie eine Reihe von Videos, die GMOCCAPY in Aktion zeigen. Leider zeigen diese Videos nicht die neueste Version von GMOCCAPY, aber die Art und Weise, sie zu benutzen, ist immer noch dieselbe wie in der aktuellen Version. Ich werde die Videos so bald wie möglich aktualisieren.

Grundlegende Verwendung

https://youtu.be/O5B-s3uiI6g

Simulierte Jog-Drehgeber

https://youtu.be/ag34SGxt97o

Einstellungen Seite

https://youtu.be/AuwhSHRJoiI

Simulierte Hardware-Taste

Deutsch: https://youtu.be/DTqhY-MfzDE

Englisch: https://youtu.be/ItVWJBK9WFA

Benutzer-Registerkarten

https://youtu.be/rG1zmeqXyZI

Videos zur Werkzeugvermessung

Simulation der automatischen Werkzeugmessung: https://youtu.be/rrkMw6rUFdk

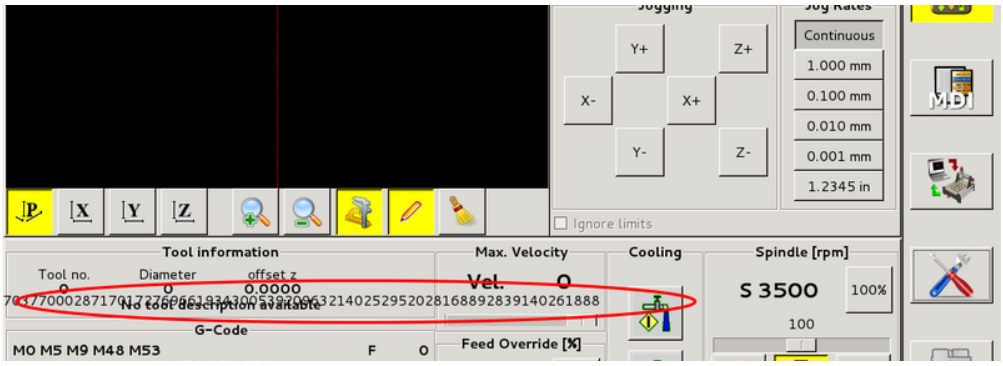
Bildschirm zur Automatischen Werkzeugmessung: https://youtu.be/Z2ULDj9dzvk

Automatische Werkzeugmessmaschine: https://youtu.be/1arucCaDdX4

10.2.12. Bekannte Probleme

Seltsame Zahlen im Infobereich

Wenn Sie im Infobereich von GMOCCAPY seltsame Zahlen erhalten wie:



Sie haben Ihre Konfigurationsdatei mit einer älteren Version von StepConfWizard erstellt. Diese hat in der INI-Datei unter [TRAJ] einen falschen Eintrag namens MAX_LINEAR_VELOCITY = xxx vorgenommen. Ändern Sie diesen Eintrag in MAX_VELOCITY = xxx.

Nicht endendes Makro

Wenn Sie ein Makro ohne Bewegung verwenden, wie dieses hier:

```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

o<zeroxy> endsub
m2
```

GMOCCAPY wird das Ende des Makros nicht sehen, weil der Interpreter seinen Zustand auf IDLE ändern muss, aber das Makro setzt den Interpreter nicht einmal in einen neuen Zustand. Um dies zu vermeiden, fügen Sie einfach eine G4 P0.1-Zeile hinzu, um das benötigte Signal zu erhalten. Das korrekte Makro würde lauten:

```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

G4 P0.1

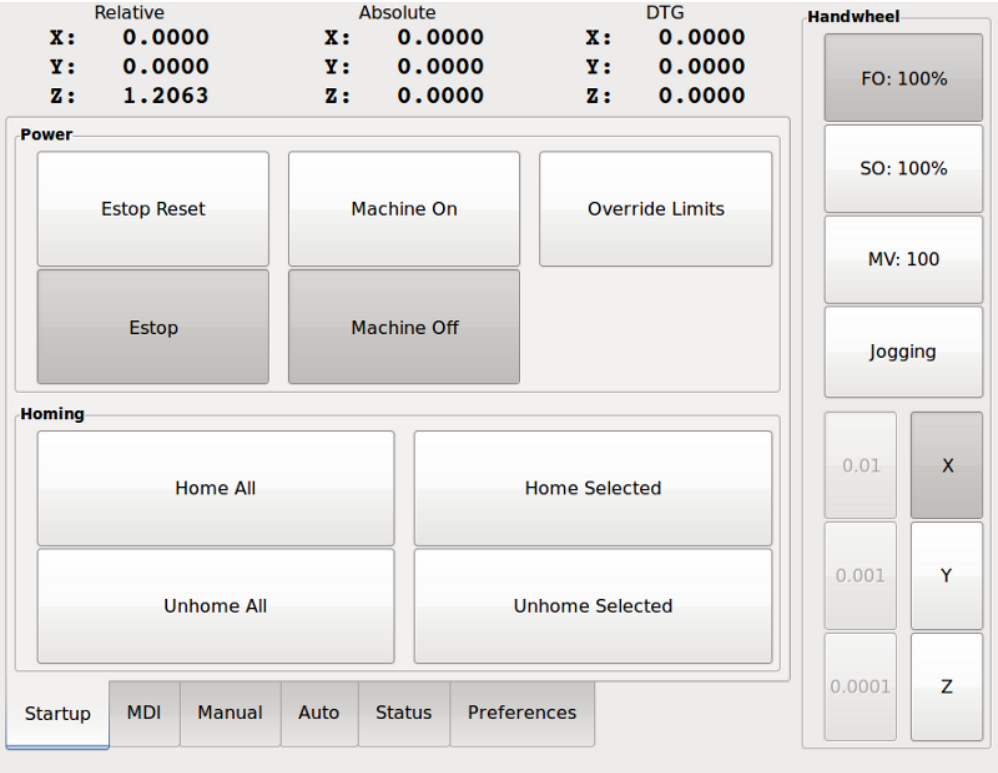
o<zeroxy> endsub
m2
```

10.3. Die Touchy Grafische Benutzeroberfläche

Touchy ist eine Benutzeroberfläche für LinuxCNC, die für die Verwendung auf Maschinenbedienfeldern gedacht ist und daher keine Tastatur oder Maus benötigt.

Es ist für die Verwendung mit einem Touchscreen gedacht und funktioniert in Kombination mit einem Rad/MPG und einigen Tasten und Schaltern.

Die Registerkarte "Handrad" verfügt über Optionsfelder zur Auswahl zwischen den Funktionen "Vorschub-Override", "Spindel-Override", "Maximale Geschwindigkeit" und "Tippen" für den Rad/MPG-Eingang. Optionsfelder für die Achsenauswahl und die Schrittweite für den Tippbetrieb sind ebenfalls vorhanden.



10.3.1. Panel-Konfiguration

HAL-Verbindungen

Touchy sucht in der INI-Datei unter der Überschrift *[HAL]* nach Einträgen von *POSTGUI_HALFILE=<Dateiname>*. Typischerweise ist *<Dateiname> touchy_postgui.hal*, kann aber ein beliebiger legaler Dateiname sein. Diese Befehle werden nach der Erstellung des Bildschirms ausgeführt, um sicherzustellen, dass die HAL-Pins des Widgets verfügbar sind. Sie können mehrere Zeilen von *POSTGUI_HALFILE=<Dateiname>* in der INI haben. Sie werden nacheinander in der Reihenfolge ausgeführt, in der sie in der INI-Datei erscheinen.

NOTE Touchy verlangte bisher, dass Sie eine Datei namens "touchy.hal" in Ihrem Konfigurationsverzeichnis (dem Verzeichnis, in dem sich Ihre INI-Datei befindet) erstellen. Aus Legacy-Gründen wird dies auch weiterhin funktionieren, aber INI-basierte Postgui-Dateien sind vorzuziehen.

Für weitere Informationen über HAL-Dateien und den net-Befehl siehe HAL Basics.

Touchy hat mehrere Ausgangspins, die mit dem Motion Controller verbunden werden sollen, um das Joggen der Räder zu steuern:

- *touchy.jog.wheel.increment*, das mit dem Pin *axis.N.jog-scale* jeder Achse *N* verbunden werden soll.
- *touchy.jog.wheel.N*, das für jede Achse *N* mit *axis.N.jog-enable* verbunden werden muss.

NOTE *N* steht für die Achsennummer 0-8.

- Zusätzlich zur Verbindung mit `touchy.wheel-counts` sollten die Radzählungen auch mit *axis.N.jog-*

counts für jede Achse *N* verbunden werden. Wenn Sie die HAL-Komponente *ilowpass* zur Glättung der Radbewegung verwenden, stellen Sie sicher, dass Sie nur *axis.N.jog-counts* und nicht `touchy.wheel-counts` glätten.

Erforderliche Kontrollen

- Abbruch-Button (Moment-Kontakt), angeschlossen an den HAL-Pin *touchy.abort*.
- Button für Zyklusstart (Moment-Kontakt) angeschlossen an *touchy.cycle-start*.
- Rad/MPG, verbunden mit *touchy.wheel-counts* und Motion Pins wie oben beschrieben.
- Einzelner Block (Kippschalter), angeschlossen an *touchy.single-block*.

Optionale Bedienelemente

- Für kontinuierliches Joggen ein bidirektionaler Momentan-Schalter (oder zwei momentane Tasten) für jede Achse, eingehängt an *touchy.jog.continuous.x.negative*, *touchy.jog.continuous.x.positive* usw.
- Wenn ein Quill-Up-Button gewünscht wird (um Z mit Höchstgeschwindigkeit an die Spitze der Reise zu joggen), ist eine momentane Taste mit *touchy.quill-up* verbunden.

Optionale Panel-Leuchten

- *touchy.jog.active* zeigt an, wenn die Panel-Jogging-Steuerelemente live sind.
- touchy.status-indicator' leuchtet, wenn die Maschine G-Code ausführt, und blinkt, wenn die Maschine zwar ausgeführt wird, sich aber in Pause/Feedhold befindet.

Empfohlen für jede Einrichtung

- Notaus-Button fest in der Notaus-Kette verdrahtet

10.3.2. Einrichtung

Touchy aktivieren

Um Touchy zu verwenden, ändern Sie im Abschnitt *[DISPLAY]* Ihrer INI-Datei die Zeile für die Anzeigeauswahl in *DISPLAY* = *touchy*.

Einstellungen

Wenn Sie Touchy zum ersten Mal starten, überprüfen Sie die Registerkarte "Einstellungen". Wenn Sie einen Touchscreen verwenden, wählen Sie die Option zum Ausblenden des Zeigers, um optimale Ergebnisse zu erzielen.

Das Statusfenster hat eine feste Höhe, die durch die Größe einer festen Schriftart bestimmt wird. Dies kann durch die Gnome-DPI beeinflusst werden, die unter System / Einstellungen / Aussehen / Schriftarten / Details eingestellt wird. Wenn der untere Teil des Bildschirms abgeschnitten ist, verringern Sie die DPI-Einstellung.

Alle anderen Schriftgrößen können auf der Registerkarte Voreinstellungen geändert werden.

Fitting the screen

Touchy bounds its window to the monitor and scrolls any tab whose content is larger than the screen, so the window never grows past the display edge (for example after loading a large tool table). The handwheel column is hidden on the Preferences tab so the settings have the full width.

If the interface is still too large for a small screen, Touchy offers, once, to shrink the display fonts to fit:

- **Shrink to fit and save** scales the fonts to the largest size that fits and saves them as your preference (the font selectors update to match).
- **Not now** leaves the fonts unchanged and does not ask again for this screen size; it is offered again only if you move to a smaller screen.
- **Never ask again** disables the offer on every screen.

To turn the offer back on, tick **Offer to shrink fonts to fit the screen** in Preferences / Display Options, or set `fit_fonts = ask` in `~/.touchy_preferences` (the **Never ask again** choice stores `fit_fonts = never`).

Makros

Touchy kann O-Wort-Makros über die MDI-Schnittstelle aufrufen. Um dies zu konfigurieren, fügen Sie im Abschnitt *[MACROS]* der INI-Datei eine oder mehrere *MACRO*-Zeilen ein. Jede Zeile sollte das folgende Format haben:

```
MACRO=increment xinc yinc
```

In diesem Beispiel ist "increment" der Name des Makros, das zwei Parameter namens xinc und yinc akzeptiert.

Legen Sie nun das Makro in einer Datei mit dem Namen "increment.ngc" im Verzeichnis "PROGRAM_PREFIX" oder einem beliebigen Verzeichnis im "SUBROUTINE_PATH" ab.

Es sollte wie folgt aussehen:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Beachten Sie, dass der Name des Unterprogramms exakt mit dem Dateinamen und dem Makronamen übereinstimmt, einschließlich Groß- und Kleinschreibung.

Wenn Sie das Makro durch Drücken der Schaltfläche Makro auf der Registerkarte MDI in Touchy aufrufen, können Sie Werte für xinc und yinc eingeben. Diese werden als *#1* bzw. *#2* an das Makro übergeben. Parameter, die Sie leer lassen, werden als Wert 0 übergeben.

Wenn es mehrere verschiedene Makros gibt, drücken Sie wiederholt die Makrotaste, um sie zu durchlaufen.

Wenn Sie in diesem einfachen Beispiel -1 für xinc eingeben und den Zyklusstart drücken, wird eine schnelle *G0*-Bewegung ausgelöst, die eine Einheit nach links geht.

Diese Makrofunktion ist nützlich für das Antasten von Kanten/Löchern und andere Einrichtungsaufgaben sowie vielleicht für das Fräsen von Löchern oder andere einfache Operationen, die vom Bedienfeld aus durchgeführt werden können, ohne dass speziell geschriebene G-Code-Programme erforderlich sind.

10.4. Gscreen

10.4.1. Einführung

Gscreen ist eine Infrastruktur zur Anzeige eines benutzerdefinierten Bildschirms zur Steuerung von LinuxCNC. Gscreen lehnt sich stark an GladeVCP an. GladeVCP verwendet den GTK-Widget-Editor GLADE, um virtuelle Bedienfelder (VCP) durch Zeigen und Klicken zu erstellen. Gscreen kombiniert dies mit Python-Programmierung, um einen GUI-Bildschirm für den Betrieb einer CNC-Maschine zu erstellen.

Gscreen ist anpassbar, wenn Sie verschiedene Tasten und Status-LEDs wünschen. Gscreen unterstützt GladeVCP, das zum Hinzufügen von Steuerelementen und Anzeigen verwendet wird. Zum Anpassen von Gscreen verwenden Sie den Glade-Editor. Gscreen ist nicht auf das Hinzufügen eines benutzerdefinierten Panels auf der rechten Seite oder einer benutzerdefinierten Registerkarte beschränkt, sondern kann vollständig bearbeitet werden.

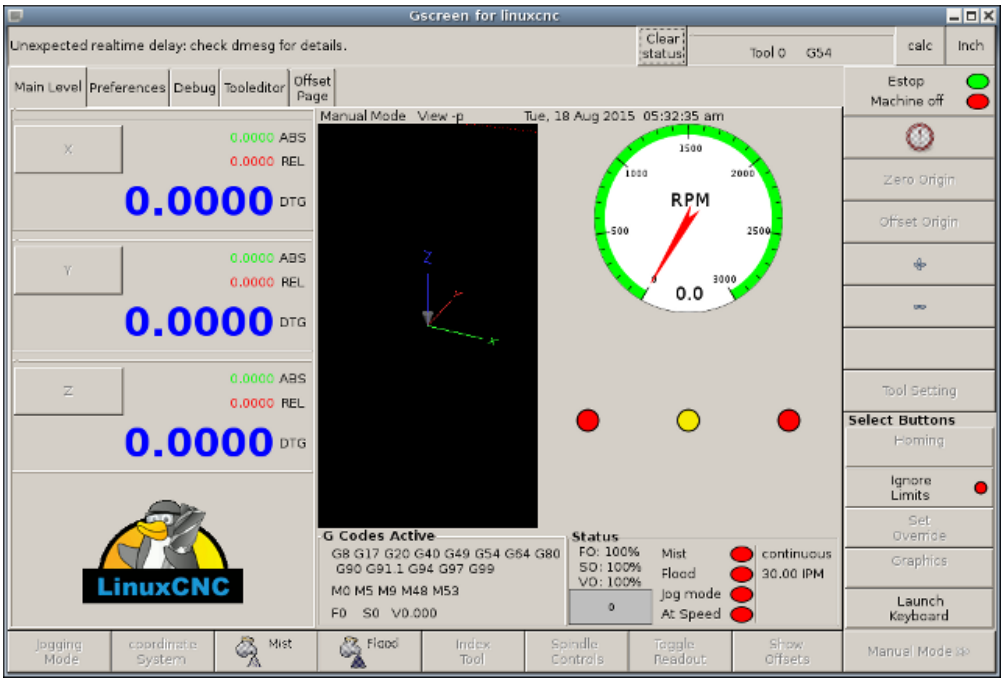


Figure 170. Gscreen Standardbildschirm

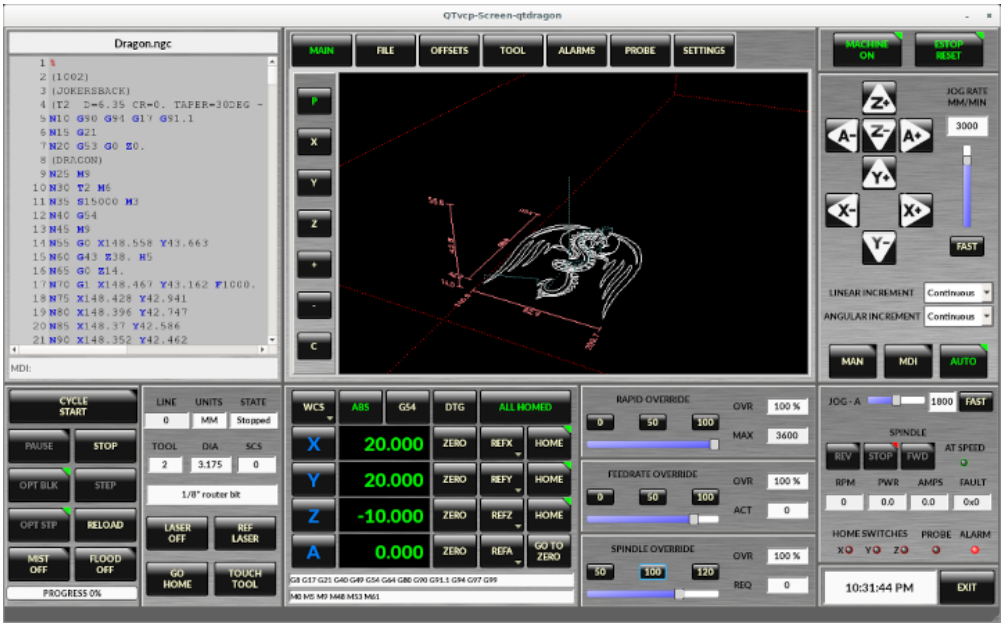


Figure 171. Gscreen Silverdragon-Bildschirm

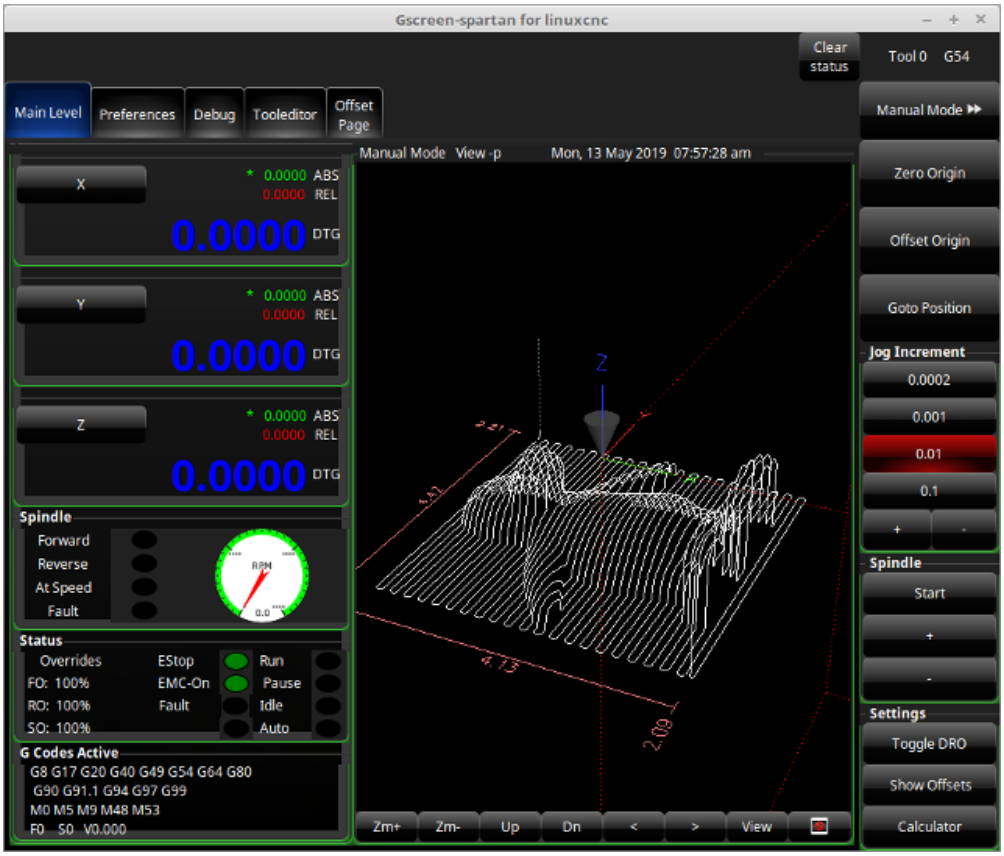


Figure 172. Gscreen Spartan-Bildschirm

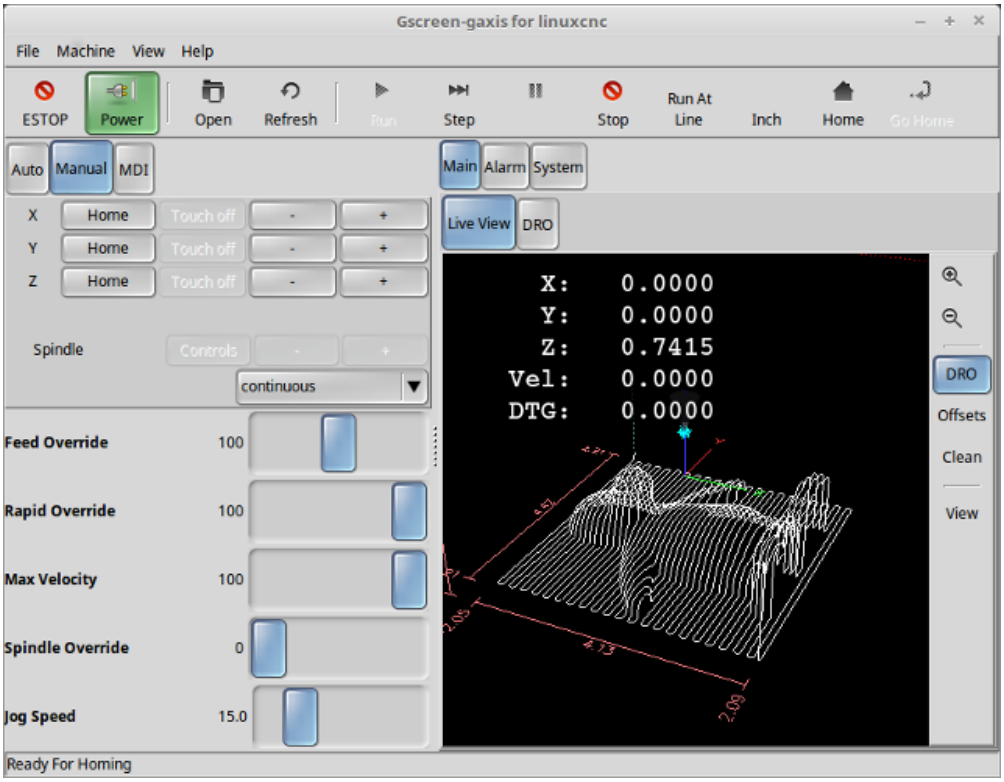


Figure 173. Gscreen Gaxis-Bildschirm



Figure 174. Gscreen Industrieller Bildschirm

Gscreen basiert auf *Glade* (dem Editor), *PyGTK* (dem Widget-Toolkit) und *GladeVCP* (die Verbindung von LinuxCNC zu Glade und PyGTK). GladeVCP hat einige spezielle Widgets und Aktionen, die nur für LinuxCNC hinzugefügt wurden. Ein Widget ist nur der allgemeine Name, der für die Buttons, Schieberegler, Beschriftungen usw. des PyGTK-Toolkits verwendet wird.

Glade-Datei

Eine Glade-Datei ist eine Textdatei, die im XML-Standard organisiert ist und das Layout und die Widgets des Bildschirms beschreibt. PyGTK verwendet diese Datei, um diese Widgets anzuzeigen und darauf zu reagieren. Der Glade-Editor macht es relativ einfach, diese Datei zu erstellen und zu bearbeiten. Sie müssen den Glade-Editor 3.38.2 verwenden, der die GTK3-Widgets verwendet.

PyGTK

PyGTK ist die Python-Bindung an GTK. GTK ist das *Toolkit* von visuellen Widgets, es ist in C programmiert. PyGTK verwendet Python, um sich mit GTK zu *binden*.

10.4.2. GladeVCP

GladeVCP verbindet LinuxCNC, HAL, PyGTK und Glade miteinander. LinuxCNC benötigt einige spezielle Widgets und GladeVCP liefert diese. Viele sind nur HAL-Erweiterungen zu bestehenden PyGTK-Widgets. GladeVCP erzeugt die HAL-Pins für die speziellen Widgets, die in der Glade-Datei beschrieben sind. GladeVCP erlaubt es auch, Python-Befehle hinzuzufügen, um mit den Widgets zu interagieren und sie Dinge tun zu lassen, die in ihrer Standardform nicht verfügbar sind. Wenn Sie ein GladeVCP-Panel bauen können, dann können Sie auch Gscreen anpassen!

Übersicht

Es gibt zwei Dateien, die einzeln oder in Kombination verwendet werden können, um Anpassungen vorzunehmen. Lokale Glade-Dateien und Handler-Dateien. Normalerweise verwendet Gscreen die Standard-Gladedatei und möglicherweise eine Handler-Datei (bei Verwendung eines Beispiel-"Skins"). Sie können Gscreen so einstellen, dass es "lokale" Glade- und Handler-Dateien verwendet. Gscreen sucht in dem Ordner, der alle Konfigurationsdateien für die von Ihnen gewählte Konfiguration enthält.

Lokale Glade-Dateien

Wenn vorhanden, werden lokale Glade-Dateien im Konfigurationsordner anstelle der Standard-Gladedateien geladen. Lokale Glade-Dateien ermöglichen es Ihnen, Ihre eigenen Designs anstelle der Standardbildschirme zu verwenden. Es gibt einen Schalter in der INI-Datei, um den Basisnamen festzulegen: -c name, damit Gscreen nach MYNAME.glade und MYNAME_handler.py sucht.

Sie können Gscreen anweisen, nur die Glade-Datei zu laden und seine internen Signale nicht mit ihr zu verbinden. Dies erlaubt gscreen, jede vom GTK-Builder gespeicherte Glade-Datei zu laden. Das bedeutet, dass Sie einen komplett benutzerdefinierten Bildschirm anzeigen können, aber auch, dass Sie eine Handler-Datei verwenden müssen. Gscreen verwendet die Glade-Datei, um die Widgets zu definieren, damit es sie anzeigen und mit ihnen interagieren kann. Viele von ihnen haben spezifische Namen, anderen hat Glade generische Namen gegeben. Wenn das Widget angezeigt, aber nie verändert wird, ist ein allgemeiner Name in Ordnung. Wenn man das Widget steuern oder mit ihm interagieren muss, wird ein hoffentlich zweckmäßiger Name vergeben (alle Namen müssen eindeutig sein). Für Widgets können im GLADE-Editor auch Signale definiert werden. Hier wird festgelegt, welches Signal gegeben wird und welche Methode aufgerufen werden soll.

Ändern von Standard-Skins

Wenn Sie den Namen eines Widgets ändern, kann es sein, dass Gscreen es nicht finden kann. Wenn auf

dieses Widget im Python-Code verwiesen wird, funktioniert das Widget im besten Fall nicht mehr, im schlimmsten Fall führt es zu einem Fehler beim Laden. Die Standardbildschirme von Gscreen verwenden nicht viele Signale, die im Editor definiert sind, sondern im Python-Code. Wenn Sie ein Widget mit Signalen verschieben (ausschneiden und einfügen), werden die Signale nicht kopiert. Sie müssen sie manuell wieder hinzufügen.

Handler-Dateien

Eine Handler-Datei ist eine Datei, die Python-Code enthält, den Gscreen zu seinen Standardroutinen hinzufügt. Eine Handler-Datei ermöglicht es, Standardeinstellungen zu ändern oder einem Gscreen-Skin Logik hinzuzufügen, ohne Gscreen selbst ändern zu müssen. Sie können neue Funktionen mit den Funktionen von Gscreen kombinieren, um das Verhalten nach Belieben zu ändern. Sie können alle Funktionen von Gscreen komplett umgehen und sie völlig anders arbeiten lassen. Wenn eine Handler-Datei mit dem Namen gscreen_handler.py (oder MYNAME_handler.py, wenn Sie den INI-Schalter verwenden) vorhanden ist, wird diese geladen, und wenn nur eine Datei registriert ist, sucht Gscreen nach der Handler-Datei; wenn sie gefunden wird, sucht sie nach bestimmten Funktionsnamen und ruft diese anstelle der Standardfunktionen auf. Wenn Sie Widgets hinzufügen, können Sie Signalaufrufe aus dem Glade-Editor einrichten, um Routinen aufzurufen, die Sie in der Handler-Datei geschrieben haben. Auf diese Weise können Sie benutzerdefiniertes Verhalten haben. Handler-Routinen können die Standardroutinen von Gscreen aufrufen, entweder vor oder nach der Ausführung ihres eigenen Codes. Auf diese Weise können Sie zusätzliches Verhalten, wie z.B. das Hinzufügen eines Tons, einbauen. Bitte lesen Sie das GladeVCP Kapitel für die Grundlagen der GladeVCP Handler-Dateien. Gscreen verwendet eine sehr ähnliche Technik.

Themen

Gscreen verwendet das PyGTK-Toolkit zur Anzeige des Bildschirms. PyGTK ist die Python-Sprachbindung an GTK. GTK unterstützt "Themen". Themes sind eine Möglichkeit, das Aussehen der Widgets auf dem Bildschirm zu verändern. Zum Beispiel kann die Farbe oder Größe von Schaltflächen und Schiebereglern mit Themen geändert werden. Es gibt viele GTK-Themen im Internet. Themes können auch angepasst werden, um das Erscheinungsbild bestimmter benannter Widgets zu verändern. Dies bindet die Themendatei enger an die Glade-Datei. Einige der Beispielscreen-Skins erlauben es dem Benutzer, ein beliebiges Thema auf dem System auszuwählen. Das Beispiel *gscreen* ist ein Beispiel dafür. Andere laden das Thema, das den gleichen Namen in der Konfigurationsdatei hat. Das Beispiel *gscreen-gaxis* ist ein Beispiel dafür. Dazu wird der Theme-Ordner in den Konfigurationsordner mit den INI- und HAL-Dateien gelegt und benannt: SCREENNAME_theme (SCREENNAME ist der Basisname der Dateien, z. B. gaxis_theme). Innerhalb dieses Ordners befindet sich ein weiterer Ordner namens gtk-2.0, in dem sich die Themadateien befinden. Wenn Sie diese Datei hinzufügen, wird Gscreen beim Starten standardmäßig dieses Thema verwenden. gscreen-gaxis enthält ein Beispiel für ein benutzerdefiniertes Thema, das nach bestimmten benannten Widgets sucht und das visuelle Verhalten dieser spezifischen Widgets ändert. Die Schaltflächen "Estop" und "Maschine ein" verwenden andere Farben als die übrigen Schaltflächen, damit sie sich abheben. Dies geschieht in der Handler-Datei, indem man ihnen bestimmte Namen gibt und indem man bestimmte Befehle in der gtkrc-Datei des Themas hinzufügt. Für einige Informationen über GTK-Themen (das Beispielthema verwendet die Pixmap-Themen-Engine), siehe: GTK-Themen, Pixmap-Themen-Engine.

Ein GladeVCP-Panel erstellen

Gscreen ist nur ein großes, kompliziertes GladeVCP-Panel, mit Python-Code zur Steuerung. Um es

anzupassen, müssen wir die Glade-Datei im Glade-Editor laden.

Installiertes LinuxCNC

Wenn Sie LinuxCNC 2.6+ auf Ubuntu 10.04 installiert haben, starten Sie einfach den Glade-Editor aus dem Anwendungsmenü oder über das Terminal. Neuere Versionen von Linux benötigen Sie Glade 3.8.0 - 3.8.6 zu installieren (möglicherweise müssen Sie es selbst kompilieren).

RIP-kompilierte Befehle

Mit einer aus dem Quellcode kompilierten Version von LinuxCNC öffnen Sie ein Terminal und cd zum Anfang des LinuxCNC-Ordners. Richten Sie die Umgebung ein, indem Sie . *./scripts/rip-environment* eingeben. Geben Sie nun *glade* ein, Sie sehen eine Reihe von Warnungen im Terminal, die Sie ignorieren können und der Editor sollte sich öffnen. Die Standard-Gscreen-Gladedatei befindet sich in: src/emc/usr_intf/gscreen/ Beispielskins befinden sich in /share/gscreen/skins/. Diese sollte in einen Konfigurationsordner kopiert werden. Sie können auch eine saubere Glade-Datei erstellen, indem Sie sie in einem Konfigurationsordner speichern.

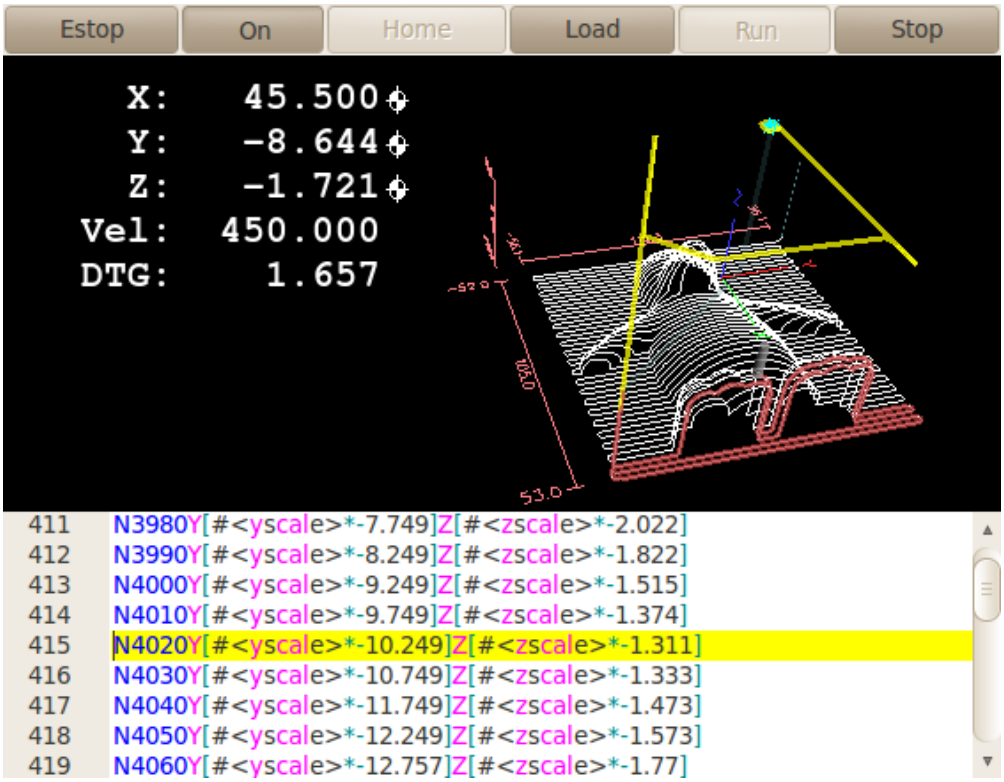
Ok, Sie haben die Glade-Datei geladen und können sie nun bearbeiten. Das erste, was Ihnen auffällt, ist, dass es im Editor nicht so aussieht, wie es angezeigt wird. Gscreen verwendet einige Tricks, wie z. B. das Ausblenden aller Schaltflächenfelder bis auf eines und das Ändern dieses Feldes je nach Modus. Dasselbe gilt für die Notizbücher. Einige Bildschirme verwenden Notizbücher, bei denen die Registerkarten nicht angezeigt werden. Um die Seiten im Editor zu wechseln, müssen Sie diese Registerkarten vorübergehend einblenden.

Wenn Sie Änderungen vornehmen, ist es viel einfacher, Widgets hinzuzufügen, als Widgets zu entfernen, damit der Bildschirm trotzdem richtig funktioniert. Das funktioniert nicht immer, manche Widgets werden wieder sichtbar gemacht. Das Ändern der Namen der regulären Widgets von Gscreen wird wahrscheinlich nicht gut funktionieren, ohne den Python-Code zu ändern, aber das Verschieben eines Widgets unter Beibehaltung des Namens ist normalerweise machbar.

Gscreen nutzt die GladeVCP-Widgets so weit wie möglich, um das Hinzufügen von Python-Code zu vermeiden. Die Kenntnis der GladeVCP-Widgets ist eine Voraussetzung. Wenn die vorhandenen Widgets die gewünschte oder benötigte Funktion bieten, muss kein Python-Code hinzugefügt werden, sondern nur die Glade-Datei im Konfigurationsordner gespeichert werden. Wenn Sie etwas benutzerdefinierteres benötigen, müssen Sie etwas Python-Programmierung vornehmen. Der Name des übergeordneten Fensters muss window1 lauten. Gscreen nimmt diesen Namen an.

Denken Sie daran, wenn Sie eine benutzerdefinierte Bildschirmoption verwenden, sind SIE dafür verantwortlich, diese zu reparieren (falls erforderlich), wenn Sie LinuxCNC aktualisieren.

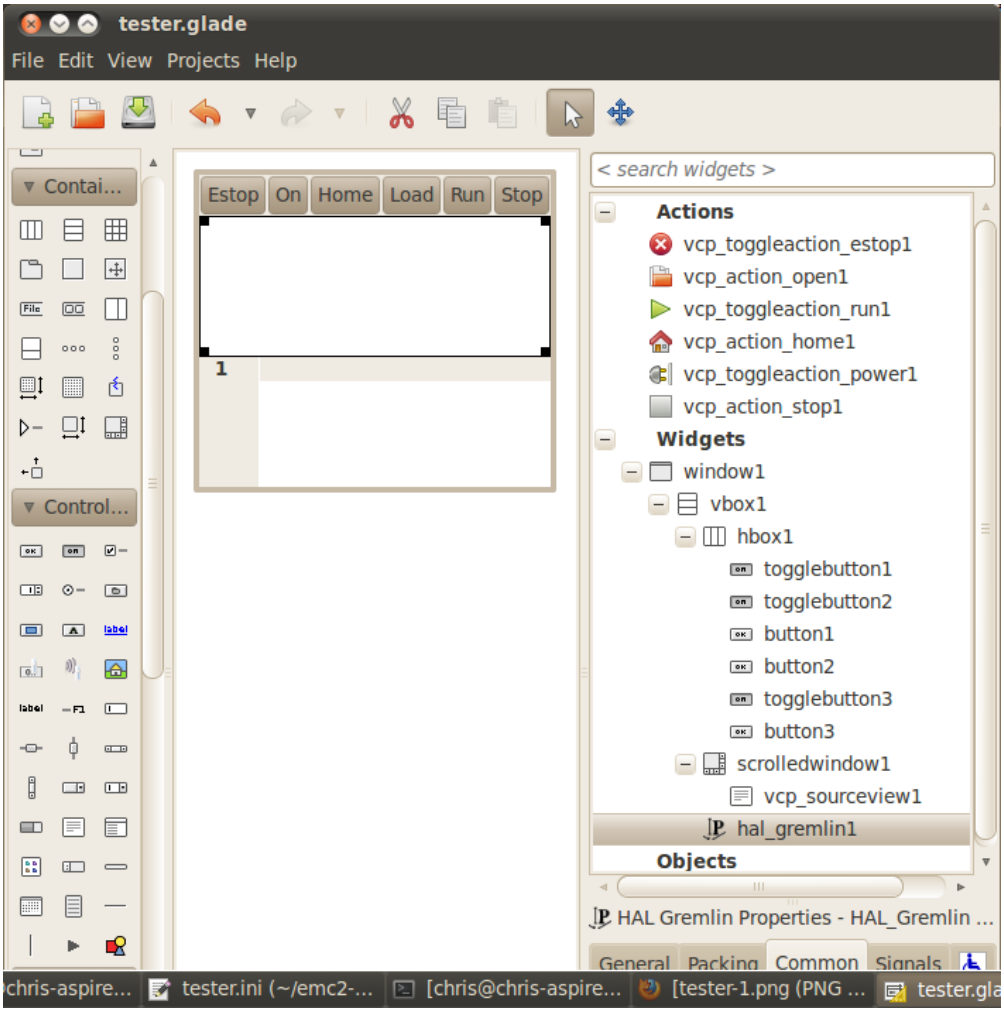
10.4.3. Erstellen eines einfachen benutzerdefinierten Bildschirms



Lassen Sie uns einen einfachen brauchbaren Bildschirm erstellen. Erstellen Sie diesen im Glade-Editor (wenn Sie ein RIP-Paket verwenden, führen Sie ihn von einem Terminal aus, nachdem Sie . scripts/rip-environment verwendet haben).

Zu beachtende Punkte:

- Das Fenster der obersten Ebene muss den Standardnamen "window1" tragen - Gscreen verlässt sich auf diesen Namen.
- Fügen Sie Aktionen hinzu, indem Sie mit der rechten Maustaste klicken und "Als Toplevel-Widget hinzufügen" wählen. Sie fügen dem Fenster nichts Visuelles hinzu, sondern werden der rechten Aktionsliste hinzugefügt. Fügen Sie alle Aktionen hinzu, die Sie oben rechts sehen.
- Nach dem Hinzufügen der Aktionen müssen wir die Schaltflächen mit den Aktionen verknüpfen, damit sie funktionieren (siehe unten).
- Das Gremlin-Widget hat keine Standardgröße, daher ist die Angabe einer gewünschten Größe hilfreich (siehe unten).
- Das Sourceview-Widget wird versuchen, das gesamte Fenster zu verwenden, so dass das Hinzufügen zu einem gescrollten Fenster dies abdeckt. (Dies wurde bereits im Beispiel getan.)
- Die Schaltflächen werden sich ausdehnen, wenn das Fenster vergrößert wird, was unschön ist, also werden wir das Feld, in dem sie sich befinden, so einstellen, dass es sich nicht ausdehnt (siehe unten).
- Die zu verwendenden Button-Typen hängen von der verwendeten VCP_action ab -eg vcp_toggle_action erfordern in der Regel Toggle-Schaltflächen (folgen Sie zunächst dem Beispiel).
- Die Tasten in diesem Beispiel sind normale Tasten und keine HAL-Buttons. Wir brauchen die HAL-Pins nicht.



In diesem Bildschirm verwenden wir VCP_actions, um LinuxCNC die Aktionen, die wir wollen, zu kommunizieren. Dies ermöglicht es uns, Standard-Funktionen ohne Hinzufügen von Python-Code in der Handler-Datei. Verknüpfen wir den Toggle-Estop-Knopf mit der Estop-Aktion Wählen Sie den Toggle-Estop-Knopf und suchen Sie unter der Registerkarte "Allgemein" nach "Related Action" und klicken Sie auf die Schaltfläche daneben. Wählen Sie nun die Aktion zum Umschalten der Sperrung aus. Jetzt schaltet die Schaltfläche die Sperrung ein und aus, wenn sie angeklickt wird. Auf der Registerkarte "Allgemein" können Sie den Text der Schaltflächenbeschriftung ändern, um die Funktion der Schaltfläche zu beschreiben. Tun Sie dies für alle Schaltflächen.

Wählen Sie das Gremlin-Widget aus, klicken Sie auf die Registerkarte Allgemein, setzen Sie die gewünschte Höhe auf 100 und klicken Sie auf das Kontrollkästchen daneben.

Klicken Sie auf das horizontale Feld, in dem sich die Schaltflächen befinden. Klicken Sie auf die Registerkarte "Verpackung" (engl. packing) und klicken Sie bei "Erweitern" (engl. expand) auf "Nein".

Speichern Sie es als tester.glade und speichern Sie es in sim/gscreen/gscreen_custom/ Ordner. Nun starten Sie LinuxCNC und klicken Sie auf sim/gscreen/gscreen_custom/tester und starten Sie es. Wenn alles gut geht, wird unser Bildschirm auftauchen und die Knöpfe werden ihre Arbeit tun. Das funktioniert, weil die tester.ini gscreen anweist, nach tester.glade und tester_handler.py zu suchen und zu laden. Die Datei tester_handler.py befindet sich in diesem Ordner und ist so programmiert, dass sie nur den Bildschirm anzeigt und nicht viel mehr. Da die speziellen Widgets direkt mit LinuxCNC kommunizieren, können Sie trotzdem nützliche Dinge tun. Wenn Ihr Bedarf an Bildschirmen durch die verfügbaren speziellen Widgets abgedeckt ist, dann ist das alles, was Sie brauchen, um einen Bildschirm

zu erstellen. Wenn Sie etwas mehr wollen, gibt es immer noch viele Tricks zur Verfügung von nur Hinzufügen von "Funktionsaufrufe", um canned Verhalten zu erhalten. Sie können auch Ihren eigenen Python-Code programmieren, um genau das zu erreichen, was Sie wollen. Das bedeutet aber, dass Sie sich mit Handler-Dateien vertraut machen müssen.

10.4.4. Beispiel für eine Handler-Datei

Es gibt spezielle Funktionen, auf die Gscreen die Handler-Datei überprüft. Wenn Sie diese in Ihre Handler-Datei aufnehmen, ruft Gscreen sie anstelle der gleichnamigen internen Funktionen von Gscreen auf.

- initialize_preferences(self): Sie können neue Einstellungsroutinen installieren.
- initialize_keybindings(self) Sie können neue Tastenbindungsroutinen installieren. In den meisten Fällen werden Sie dies nicht tun wollen, sondern die einzelnen Tastaturbindungsaufrufe außer Kraft setzen wollen. Sie können auch weitere Tastenbindungen hinzufügen, die eine beliebige Funktion aufrufen.
- initialize_pins(self): erzeugt / initialisiert HAL-Pins
- connect_signals(self,handlers): Wenn Sie einen völlig anderen Bildschirm als den Standard-Gscreen verwenden, müssen Sie dies hinzufügen, da gscreen sonst versucht, Signale mit Widgets zu verbinden, die nicht vorhanden sind. Die Standardfunktion von Gscreen wird mit self.gscreen.connect_signals(handlers) aufgerufen. Wenn Sie nur zusätzliche Signale zu Ihrem Bildschirm hinzufügen möchten, aber trotzdem die Standardsignale verwenden wollen, rufen Sie zuerst diese Funktion auf und fügen dann weitere Signale hinzu. Wenn Ihre Signale einfach sind (keine Benutzerdaten übergeben), können Sie auch die Glade-Signalauswahl im Glade-Editor verwenden.
- initialize_widgets(self): Hiermit können Sie alle Widgets einrichten. Gscreen ruft normalerweise *self.gscreen.initialize_widgets()* auf, das eigentlich viele separate Funktionen aufruft. Wenn Sie einige dieser Widgets einbinden möchten, rufen Sie diese Funktionen einfach direkt auf. Oder fügen Sie self.gscreen.init_show_windows() hinzu, damit die Widgets nur angezeigt werden. Dann, falls gewünscht, initialisieren/anpassen Sie Ihre neuen Widgets.
- initialize_manual_toolchange(self): Ermöglicht eine vollständige Überarbeitung des manuellen Werkzeugwechselsystems.
- set_restart_line(self.line):
- timer_interrupt(self): ermöglicht die vollständige Neudefinition der Interrupt-Routine. Dies wird für den Aufruf von periodic() und die Überprüfung auf Fehler von linuxcnc.status verwendet.
- check_mode(self): wird verwendet, um zu prüfen, in welchem Modus sich der Bildschirm befindet. Liefert eine Liste[] 0 -manual 1- mdi 2- auto 3- jog.
- on_tool_change(self,widget): Sie können dies verwenden, um den manuellen Werkzeugwechsel-Dialog zu überschreiben - dieser wird aufgerufen, wenn *gscreen.tool-change* den Status ändert.
- dialog_return(self,dialog_widget,displaytype,pinname): Verwenden Sie diese Funktion, um eine Benutzermeldung oder einen manuellen Werkzeugwechsel-Dialog außer Kraft zu setzen. Wird aufgerufen, wenn der Dialog geschlossen wird.
- periodisch(self): Diese Funktion wird alle (standardmäßig 100) Millisekunden aufgerufen.

Verwenden Sie es, um Ihre Widgets/HAL-Pins zu aktualisieren. Sie können danach auch Gscreen regular periodic aufrufen, self.gscreen.update_position() oder einfach pass hinzufügen, um nichts zu aktualisieren. Die Funktion update_position() von Gscreen ruft eigentlich viele separate Funktionen auf. Wenn Sie einige dieser Widgets einbinden möchten, rufen Sie diese Funktionen einfach direkt auf.

Sie können auch eigene Funktionen hinzufügen, die in dieser Datei aufgerufen werden sollen. Normalerweise würden Sie einem Widget ein Signal hinzufügen, um Ihre Funktion aufzurufen.

Hinzufügen von Funktionen für Tastenkombinationen

Unser Tester-Beispiel wäre nützlicher, wenn es auf Tastaturbefehle reagieren würde. Es gibt eine Funktion namens keybindings(), die versucht, dies einzurichten. Man kann sie zwar komplett außer Kraft setzen, was wir nicht getan haben, aber sie setzt einige Dinge voraus:

- Es wird davon ausgegangen, dass die Umschalttaste für den Ausstieg *button_estop* heißt und mit der Taste F1 gesteuert wird.
- Es wird davon ausgegangen, dass der Netzschalter "button_machine_on" heißt und mit der Taste F2 gesteuert wird.

Diese lassen sich leicht beheben, indem man die Schaltflächen im Glade-Editor entsprechend umbenennt. Aber stattdessen werden wir die Standardaufrufe außer Kraft setzen und unsere eigenen hinzufügen.

Fügen Sie diese Befehle in die Handler-Datei ein:

```
# Gscreen-Funktionen überschreiben
# Tastatur-Funktionen (engl. key binding)-Aufrufe
def on_keycall_ESTOP(self,state,SHIFT,CNTRL,ALT):
  if state: # only if pressed, not released
    self.widgets.togglebutton1.emit('activate')
    self.gscreen.audio.set_sound(self.data.alert_sound)
    self.gscreen.audio.run()
    return True # stop progression of signal to other widgets
def on_keycall_POWER(self,state,SHIFT,CNTRL,ALT):
  if state:
    self.widgets.togglebutton2.emit('activate')
    return True
def on_keycall_ABORT(self,state,SHIFT,CNTRL,ALT):
  if state:
    self.widgets.button3.emit('activate')
    return True
```

Jetzt haben wir die gleichnamigen Funktionsaufrufe von Gscreen überschrieben und behandeln sie in unserer Handler-Datei. Wir verweisen jetzt auf die Widgets mit dem Namen, den wir im Glade-Editor verwendet haben. Wir haben auch eine eingebaute Gscreen-Funktion hinzugefügt, um einen Ton zu erzeugen, wenn sich Estop ändert. Beachten Sie, dass wir die eingebauten Gscreen-Funktionen mit self.gscreen.[FUNKTIONSNAME]() aufrufen müssen. Wenn wir self.[FUNKTIONSNAME]() verwenden, wird die Funktion in unserer Handler-Datei aufgerufen.

Fügen wir eine weitere Tastenkombination hinzu, die das Halmeter lädt, wenn F4 gedrückt wird.

In der Handler-Datei unter *def initialize_widgets(self):* ändern in:

```
  def initialize_widgets(self):
    self.gscreen.init_show_windows()
    self.gscreen.keylookup.add_conversion('F4','TEST','on_keycall_HALMETER')
```

Fügen Sie dann diese Funktionen unter der Klasse "HandlerClass" hinzu:

```
  def on_keycall_HALMETER(self,state,SHIFT,CNTRL,ALT):
    if state:
      self.gscreen.on_halmeter()
      return True
```

Dies fügt eine Keybinding-Konvertierung hinzu, die gscreen anweist, wenn F4 gedrückt wird on_keycall_HALMETER aufzurufen. Dann fügen wir die Funktion zur Handle-Datei hinzu, um eine Gscreen-Builtin-Funktion zum Starten von Halmeter aufzurufen.

LinuxCNC-Status Status

Das Modul *Gstat* fragt den Zustand von LinuxCNC alle 100 ms ab und sendet Callback-Nachrichten an Benutzerfunktionen, wenn sich der Zustand ändert. Sie können Nachrichten registrieren, um auf bestimmte Zustandsänderungen zu reagieren. Als Beispiel werden wir uns registrieren, um *file-loaded* -Meldungen zu erhalten, wenn LinuxCNC eine neue Datei lädt. Zuerst müssen wir das Modul importieren und instanziieren: Fügen Sie in der Import-Sektion der Handler-Datei hinzu:

```
from hal_glib import GStat
GSTAT = GStat()
```

In der Handler-Datei unter *def __init__(self):* hinzufügen:

```
GSTAT.connect('file-loaded', self.update_filepath)
```

Fügen Sie dann in der *HandlerClass* folgende Funktion hinzu:

```
self.update_filepath(self, obj, path):
    self.widgets.my_path_label.set_text(path)
```

Wenn LinuxCNC eine neue Datei lädt, sendet Gstat eine Callback-Nachricht an die Funktion *update_filepath*. In diesem Beispiel aktualisieren wir ein Label mit dem Pfadnamen (vorausgesetzt, es gibt ein Label mit dem Namen *my_path_label*) in der Glade-Datei.

Jogging-Tasten

Es gibt keine speziellen Widgets für ein Bildschirm-Button-Joggen, also müssen wir es mit Python-Code tun. Fügen Sie unter der Funktion connect_signals folgenden Code hinzu:

```
        for i in('x','y','z'):
            self.widgets[i+'neg'].connect("pressed", self['jog_'+i],0,True)
            self.widgets[i+'neg'].connect("released", self['jog_'+i],0,False)
            self.widgets[i+'pos'].connect("pressed", self['jog_'+i],1,True)
            self.widgets[i+'pos'].connect("released", self['jog_'+i],1,False)
        self.widgets.jog_speed.connect("value_changed",self.jog_speed_changed)
```

Fügen Sie diese Funktionen unter der Klasse HandlerClass hinzu:

```
    def jog_x(self,widget,direction,state):
        self.gscreen.do_key_jog(_X,direction,state)
    def jog_y(self,widget,direction,state):
        self.gscreen.do_key_jog(_Y,direction,state)
    def jog_z(self,widget,direction,state):
        self.gscreen.do_key_jog(_Z,direction,state)
    def jog_speed_changed(self,widget,value):
        self.gscreen.set_jog_rate(absolute = value)
```

Schließlich fügen Sie der GLADE-Datei für jede Achse zwei Schaltflächen hinzu - eine für die positive und eine für die negative Richtung des Tippens. Nennen Sie diese Schaltflächen xneg, xpos, yneg, ypos bzw. zneg, zpos. Fügen Sie ein SpeedControl-Widget in die GLADE-Datei ein und nennen Sie es jog_speed.

10.4.5. Gscreen Start

Gscreen ist wirklich nur eine Infrastruktur, um eine benutzerdefinierte GladeVCP-Datei zu laden und damit zu interagieren.

1. Gscreen liest die Optionen, mit denen es gestartet wurde.
2. Gscreen stellt den Debug-Modus ein und setzt den optionalen Skin-Namen.
3. Gscreen prüft, ob im Konfigurationsordner "lokale" XML-, Handler- und/oder Locale-Dateien vorhanden sind. Diese werden dann anstelle der Standarddateien (in share/gscreen/skins/) verwendet (es können zwei verschiedene Bildschirme angezeigt werden).
4. Der Hauptbildschirm wird geladen und die Übersetzungen werden eingerichtet. Falls vorhanden, wird der zweite Bildschirm geladen und die Übersetzungen werden eingerichtet.
5. Optionales Audio wird initialisiert, falls vorhanden.
6. Es liest einen Teil der INI-Datei, um die Einheiten und die Anzahl/Typen der Achsen zu initialisieren.
7. Initialisiert die Bindung von Python an HAL, um eine Nicht-Echtzeit-Komponente mit dem Namen Gscreen zu erstellen.
8. GladeVCP's makepins wird aufgerufen, um die XML-Datei zu parsen, um HAL-Pins für die HAL-Widgets zu erstellen und die mit LinuxCNC verbundenen Widgets zu registrieren.
9. Prüft, ob eine "local" Handler-Datei im Konfigurationsordner vorhanden ist, oder verwendet die Standard-Handler-Datei aus dem Skin-Ordner.
10. Wenn eine Handler-Datei vorhanden ist, analysiert Gscreen diese und registriert die Funktionsaufrufe im Namensraum von Gscreen.
11. Glade gleicht/registriert alle Signalaufrufe an Funktionen in Gscreen und der Handler-Datei.

12. Gscreen prüft die INI-Datei auf den Namen einer Optionseinstellungsdatei, andernfalls verwendet es *.gscreen_preferences* =.
13. Gscreen prüft, ob ein Aufruf der Einstellungsfunktion (*initialize_preferences(self)*) in der Handler-Datei vorhanden ist, andernfalls wird die Standardfunktion von Gscreen verwendet.
14. Gscreen sucht nach der "ClassicLadder"-Echtzeit-Komponente.
15. Gscreen prüft auf das systemweite GTK-Thema.
16. Gscreen holt sich das Inkrement beim Joggen aus der INI-Datei.
17. Gscreen holt sich die Winkelschritte für das Joggen aus der INI-Datei.
18. Gscreen holt sich die Standard- und die maximale Jog-Geschwindigkeit aus der INI.
19. Gscreen sammelt die maximale Geschwindigkeit aller Achsen aus dem TRAJ-Abschnitt der INI.
20. Gscreen prüft, ob Winkelachsen vorhanden sind, und entnimmt dann die Standard- und Höchstgeschwindigkeit aus der INI-Datei.
21. Gscreen sammelt alle Override-Einstellungen aus der INI.
22. Gscreen prüft, ob es sich um eine Drehbankkonfiguration aus der INI-Datei handelt.
23. Gscreen findet den Namen der tool_table-, tool editor- und param-Datei in der INI.
24. Gscreen prüft die Handler-Datei auf die Funktion "keybindings" ("initialize_keybindings(self)") oder verwendet die von Gscreen bereitgestellte Funktion.
25. Gscreen prüft die Handler-Datei auf die Pin-Funktion (*initialize_pins(self)*) oder verwendet die regulär von Gscreen zur Verfügung gestellte.
26. Gscreen prüft die Handler-Datei auf die Funktion manual_toolchange (*initialize_manual_toolchange(self)*) oder verwendet die regulär von Gscreen zur Verfügung gestellte.
27. Gscreen überprüft die Handler-Datei auf die Funktion connect_signals (*initialize_connect_signals(self)*) oder verwendet andernfalls eine Standarddatei von Gscreen.
28. Gscreen prüft die Handler-Datei für die Widgets-Funktion (*initialize_widgets(self)*) oder verwendet die regulär von Gscreen zur Verfügung gestellte.
29. Gscreen richtet die in der INI-Datei angegebenen Meldungen ein.
30. Gscreen teilt HAL mit, dass die Gscreen-HAL-Komponente mit der Erstellung von Pins fertig ist und bereit ist. Wenn ein Terminal-Widget auf dem Bildschirm vorhanden ist, werden alle Gscreen-Pins dorthin ausgegeben.
31. Gscreen stellt die Anzeigezykluszeit auf der Grundlage der INI-Datei ein.
32. Gscreen prüft die Handler-Datei auf den Aufruf der Funktion *timer_interupt(self)*, andernfalls wird der Standardfunktionsaufruf von Gscreen verwendet.

10.4.6. INI-Einstellungen

Unter der Überschrift [DISPLAY]:

```
DISPLAY = gscreen -c tester
  options:
   -d debugging on
```

```
-v verbose debugging on
```

Mit der Option (engl. auch switch, möglicherweise in anderem Kontext dann auch fälschlich als Schalter bezeichnet) -c kann man einen "Skin" auswählen. Gscreen geht davon aus, dass die Glade-Datei und die Handler-Datei denselben Namen haben. Der optionale zweite Bildschirm ist derselbe Name mit einer 2 (z.B. tester2.glade). Es ist keine zweite Handler-Datei erlaubt. Sie wird nur geladen, wenn sie vorhanden ist. Gscreen sucht in der LinuxCNC-Konfigurationsdatei, die zuerst gestartet wurde, nach den Dateien, dann im System-Skin-Ordner.

10.4.7. Benutzerdialog-Meldungen

Diese Funktion wird verwendet, um Pop-up-Dialogmeldungen auf dem Bildschirm anzuzeigen. Diese werden in der INI-Datei definiert und über HAL-Pins gesteuert:

MESSAGE_BOLDTEXT

ist im Allgemeinen ein Titel.

MESSAGE_TEXT

ist darunter und in der Regel länger.

MESSAGE_DETAILS

ist ausgeblendet, wenn nicht darauf geklickt wird.

MESSAGE_PINNAME

ist der Basisname der HAL-Pins.

MESSAGE_TYPE

gibt an, ob es sich um eine Ja/Nein-, eine Ok- oder eine Statusmeldung handelt

- Statusmeldungen
  - wird in der Statusleiste und im Benachrichtigungsdialog angezeigt,
  - erfordern keinen Benutzereingriff.
- OK-Meldungen
  - den Benutzer auffordern, auf ok zu klicken, um den Dialog zu schließen.
  - einen HAL-Pin haben, um den Dialog zu starten, und einen, um zu signalisieren, dass er auf eine Antwort wartet.
- Ja/Nein-Meldungen
  - den Benutzer auffordern, die Schaltflächen "Ja" oder "Nein" auszuwählen, um den Dialog zu schließen.
  - haben drei HAL-Pins:
    1. eine, um den Dialog anzuzeigen,
    2. eine für das Warten, und
    3. eine für die Antwort.

Hier ist ein Beispiel für einen INI-Code. Er befindet sich unter der Überschrift [DISPLAY].

```
# Dies wird nur in der Statusleiste und im Popup-Fenster für Desktop-Benachrichtigungen
angezeigt.
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a statusbar test
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest

# Es wird ein Dialog mit einer Ja-Nein-Frage eingeblendet
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_DETAILS = Y/N DETAILS
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yndialogtest

# Es erscheint ein Dialog, der eine OK-Antwort erfordert und in der Statusleiste und
# dem Desktop-Benachrichtigungs-Popup.
MESSAGE_BOLDTEXT = This is the short text
MESSAGE_TEXT = This is the longer text of the both type test. It can be longer then the
status bar text
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog status
MESSAGE_PINNAME = bothtest
```

Kopieren Sie die Datei "Stock Handler/Glade" zur Bearbeitung

Wenn Sie einen Standardbildschirm verwenden, aber dessen Handler-Datei ändern möchten, müssen Sie die Standarddatei in Ihren Konfigurationsdateiordner kopieren. Gscreen wird dies erkennen und die kopierte Datei verwenden. Aber wo ist die Originaldatei? Wenn Sie ein RIP LinuxCNC verwenden, befinden sich die Beispiel-Skins in /share/gscreen/skins/*SCREENNAME* Installierte Versionen von LinuxCNC haben sie an leicht unterschiedlichen Orten, je nach verwendeter Distribution. Eine einfache Möglichkeit, den Speicherort zu finden, besteht darin, ein Terminal zu öffnen und den gewünschten Sim-Screen zu starten. Im Terminal werden die Speicherorte der Dateien ausgegeben. Es kann hilfreich sein, den Schalter -d in die Zeile gscreen load in der INI einzufügen.

Hier ist ein Beispiel:

```
chris@chris-ThinkPad-T500 ~/emc-dev/src $ linuxcnc
LINUXCNC - 2.7.14
Machine configuration directory is '/home/chris/emc-
dev/configs/sim/gscreen/gscreen_custom'
Machine configuration file is 'industrial_lathe.ini'
Starting LinuxCNC...
Found file(lib): /home/chris/emc-dev/lib/hallib/core_sim.hal
Note: Using POSIX non-realtime
Found file(lib): /home/chris/emc-dev/lib/hallib/sim_spindle_encoder.hal
Found file(lib): /home/chris/emc-dev/lib/hallib/axis_manualtoolchange.hal
Found file(lib): /home/chris/emc-dev/lib/hallib/simulated_home.hal
**** GSCREEN WARNING: no audio alerts available - Is python-gst0.10 library installed?
**** GSCREEN INFO ini: /home/chris/emc-
dev/configs/sim/gscreen/gscreen_custom/industrial_lathe.ini
```

```
**** GSCREEN INFO: Skin name = industrial

**** GSCREEN INFO:  Using SKIN glade file from /home/chris/emc-
dev/share/gscreen/skins/industrial/industrial.glade ****

**** GSCREEN INFO:  No Screen 2 glade file present
**** GSCREEN INFO: handler file path: ['/home/chris/emc-
dev/share/gscreen/skins/industrial/industrial_handler.py']
```

Die Zeile:

```
**** GSCREEN INFO: handler file path: ['/home/chris/emc-
dev/share/gscreen/skins/industrial/industrial_handler.py']
```

zeigt, wo sich die Bestandsdatei befindet. Kopieren Sie diese Datei in Ihren Konfigurationsordner. Das Gleiche gilt für die Glade-Datei.

10.5. QtDragon GUI

10.5.1. Einführung

QtDragon und QtDragon_hd werden mit dem QtVCP-Framework entwickelt. Es ist die kreative Vision der Forum Persönlichkeit Persei8. Vieles davon basiert auf der hervorragenden Arbeit anderer in der LinuxCNC-Gemeinschaft. LinuxCNC's Version ist von Persei8's Github Versionen angepasst. Es ist in erster Linie für 3-5-Achsen-Maschinen wie Fräsmaschinen oder Router gedacht. Es funktioniert gut mit einem Touchscreen und/oder einer Maus. QtDragon unterstützt mehrere Möglichkeiten zum Antasten von Werkzeugen und zum Antasten von Werkstücken. Sie können LinuxCNC's externe Offsets Fähigkeit verwenden, um automatisch die Spindel während einer Pause zu erhöhen. Wenn Sie die VersaProbe-Option und Remap-Code können Sie automatische Werkzeuglängen-Abtastung beim Werkzeugwechsel hinzuzufügen.

NOTE	QtDragon und QtVCP sind relativ neue Programme in LinuxCNC hinzugefügt. Bugs und Unregelmäßigkeiten sind möglich. Bitte testen Sie sorgfältig, wenn Sie eine gefährliche Maschine benutzen. Bitte senden Sie Berichte an das Forum oder die Mailing Liste.

QtDragon

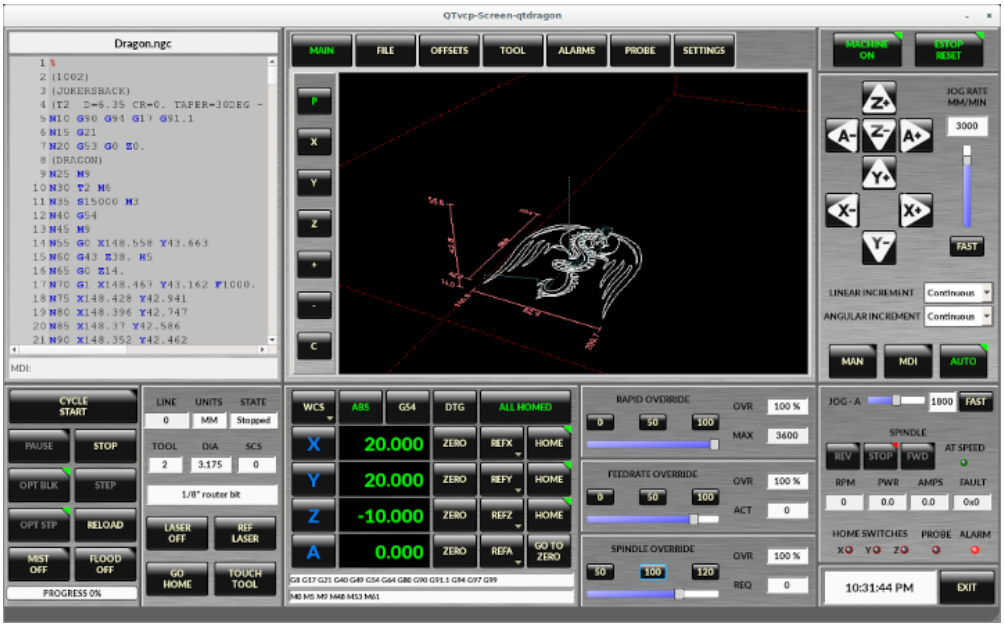


Figure 175. QtDragon - 3-5 Achsen Muster (1440x860) im Silber-look (eigentlich engl. theme)

QtDragon ist von einer Auflösung von 1280x768 bis 1680x1200 größenveränderbar. Es funktioniert im Fenstermodus auf jedem Monitor mit höherer Auflösung, aber nicht auf Monitoren mit niedrigerer Auflösung.

QtDragon_lathe



QtDragon_lathe ist eine modifizierte Version von QtDragen um den Anforderungen von Drehmaschinen besser zu genügen.
Sie ist von einer Auflösung von 1280x768 bis 1680x1200 größenveränderbar.
Sie funktioniert im Fenstermodus auf jedem Monitor mit höherer Auflösung, aber nicht auf Monitoren mit niedrigerer Auflösung.

QtDragon_hd

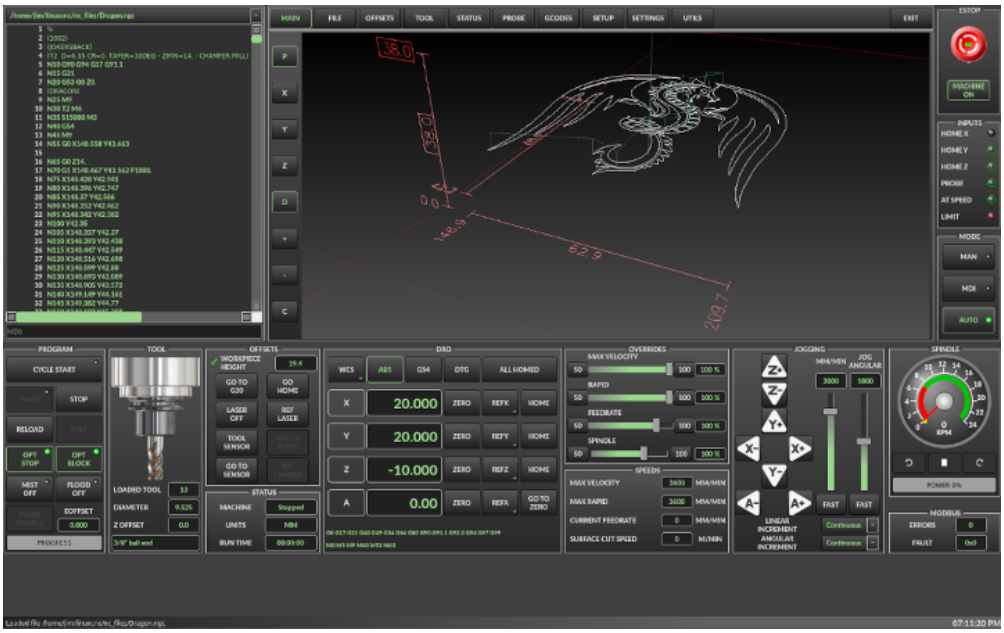


Figure 176. QtDragon_hd - 3 bis 5 Achsen-Beispiel für größere Monitore (1920x1056) im dunklen look (engl. eigentlich dark theme)

QtDragon_hd hat ein ähnliches Design wie QtDragon, wurde aber modifiziert, um den zusätzlichen Platz auf modernen, größeren Monitoren zu nutzen. Es gibt einige kleine Unterschiede im Layout und im Nutzen.

QtDragon_hd hat eine Auflösung von 1920x1056 und ist nicht größenveränderbar. Es funktioniert im Fenstermodus auf jedem Monitor mit höherer Auflösung, aber nicht auf Monitoren mit niedrigerer Auflösung.

QtDragon_hd_vertical

QtDragon_hd_vertical hat einer vertikale Orientierung. Es ist nicht größenveränderbar.

10.5.2. Erste Schritte - Die INI-Datei

Wenn Ihre Konfiguration derzeit nicht für die Verwendung von QtDragon eingerichtet ist, können Sie sie ändern, indem Sie den Abschnitten der INI-Datei bearbeiten. Eine ausführliche Liste der Optionen finden Sie im Abschnitt Display der INI-Datei-Dokumentation.

NOTE	Sie können in der INI-Datei nur einen Abschnitt (z.B. [HAL]) haben. Wenn Sie in diesen Dokumenten mehrere Abschnittsoptionen sehen, fügen Sie sie alle unter dem entsprechenden Abschnittsnamen ein.

Anzeige (engl. display)

Im Abschnitt `[DISPLAY]` ändern Sie die `DISPLAY =`-Zuweisung wie folgt:

- `qtdragon` für eine kleine Version

- `qtdradon_hd` für die große Version.

Sie können `-v`, `-d`, `-i` oder `-q` hinzufügen, um eine (entsprechend) ausführliche, debug-, info- oder stille Ausgabe auf dem Terminal zu erhalten.

```
[DISPLAY]
DISPLAY = qtvcp qtdragon
```

Einstellungen

Um den Überblick über die Einstellungen zu behalten, sucht QtDragon nach einer Einstellungs-Textdatei.
Fügen Sie den folgenden Eintrag unter der Überschrift `[DISPLAY]` hinzu.
Es kann `~` für das Heimverzeichnis (engl. home directory) oder `WORKINGFOLDER` oder `CONFIGFOLDER` verwenden, um QtVCPs zu diesen Verzeichnisse zu lenken:
Dadurch wird die Datei im config-Ordner des Startbildschirms gespeichert. (Andere Optionen sind möglich, siehe die QtVCP's screenoption Widget Dokumentation.)

```
[DISPLAY]
PREFERENCE_FILE_PATH = WORKINGFOLDER/qtdragon.pref
```

Protokollierung (engl. logging)

Sie können angeben, wo Verlauf/Protokolle gespeichert werden sollen.
Diese Dateinamen können vom Benutzer ausgewählt werden.
Fügen Sie im Abschnitt `[DISPLAY]` Folgendes hinzu:

```
[DISPLAY]
MDI_HISTORY_FILE = mdi_history.dat
MACHINE_LOG_PATH = machine_log.dat
LOG_FILE = qtdragon.log
```

Override-Kontrollen

Diese setzen die Übersteuerungsregler von qtdragon (1.0 = 100 Prozent):

```
[DISPLAY]
MAX_SPINDLE_0_OVERRIDE = 1.5
MIN_SPINDLE_0_OVERRIDE = .5
MAX_FEED_OVERRIDE       = 1.2
```

Spindelsteuerungen

Spindelsteuerungseinstellungen (in U/min und Watt):

```
[DISPLAY]
DEFAULT_SPINDLE_0_SPEED = 500
SPINDLE_INCREMENT = 200
MIN_SPINDLE_0_SPEED = 100
```

```
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_POWER = 1500
```

Jogging-Inkremente

Legen Sie wählbare Jogging-Schritte fest.
Diese Abstufungen können vom Benutzer geändert werden.

```
[DISPLAY]
INCREMENTS = Continuous, .001 mm, .01 mm, .1 mm, 1 mm, 1.0 inch, 0.1 inch, 0.01 inch
ANGULAR_INCREMENTS = 1, 5, 10, 30, 45, 90, 180, 360
```

Gitterlinien Zunahme (engl. grid increments)

Legt die verfügbaren optionalen Rastergrößen für die Grafikdarstellung fest.
Dies überschreibt die Standardgrößen.
Zur Angabe der Einheiten werden *mm* und *in* (engl. kurz für Zoll) verwendet.
Das Auswahlfeld für das Raster wird angezeigt, wenn im Grafikfenster die Taste *OPTN* angeklickt wird.

```
[DISPLAY]
GRIDS = 0, .1 mm, 1 mm, 2 mm, 5 mm, 10 mm, .25 in, .5 in
```

Jog-Geschwindigkeit

Einstellen der Jogginggeschwindigkeit (in Einheiten pro Sekunde)

```
[DISPLAY]
MIN_LINEAR_VELOCITY     = 0
MAX_LINEAR_VELOCITY     = 60.00
DEFAULT_LINEAR_VELOCITY = 50.0
DEFAULT_ANGULAR_VELOCITY = 10
MIN_ANGULAR_VELOCITY = 1
MAX_ANGULAR_VELOCITY = 360
```

Dialogsystem für Benutzermeldungen

Optionale Popup-Dialoge für benutzerdefinierte Meldungen, gesteuert durch HAL-Pins.
MESSAGE_TYPE kann *okdialog* oder *yesnodialog* sein. Siehe `qtvcp/library/messages` für weitere Informationen.
Dieses Beispiel zeigt, wie man einen Dialog erstellt, der den Benutzer auffordert, *ok* zu wählen, um ihn zu bestätigen und auszublenden.
Diese Dialoge könnten z.B. für Warnungen bei niedrigem Schmierölstand usw. verwendet werden.

```
[DISPLAY]
MESSAGE_BOLDTEXT = Dies ist der kurze Text
MESSAGE_TEXT = Dies ist der längere Text des Tests der beiden Typen. Er kann länger sein
als der Text der Statusleiste
MESSAGE_DETAILS = BOTH DETAILS
```

```
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = oktest
```

Multimessages use an sint HAL pin to pop multiple defined messages.

```
[DISPLAY]
MULTIMESSAGE_ID = VFD

MULTIMESSAGE_VFD_NUMBER = 1
MULTIMESSAGE_VFD_TYPE = okdialog status
MULTIMESSAGE_VFD_TITLE = VFD Fehler: 1
MULTIMESSAGE_VFD_TEXT = Die ist der längere Text FÜR MESSAGE NUMMER 1
MULTIMESSAGE_VFD_DETAILS = DETAILS für VFD Fehler 1
MULTIMESSAGE_VFD_ICON = WARNING

MULTIMESSAGE_VFD_NUMBER = 2
MULTIMESSAGE_VFD_TYPE = nonedialog status
MULTIMESSAGE_VFD_TITLE = VFD Fehler: 2
MULTIMESSAGE_VFD_TEXT = Die sist der längere Text FÜR MESSAGE NUMMER 2
MULTIMESSAGE_VFD_DETAILS = DETAILS für VFD Fehler 2
MULTIMESSAGE_VFD_ICON = INFO
```

Benutzerdefinierte VCP-Panels einbetten

Sie können optional QtVCP Virtual Control Panels in den QtDragon Bildschirmen einbetten.
Diese Panels können entweder vom Benutzer erstellt oder eingebaut werden QtVCP Panels.
Siehe QtVCP/VCP Panels für andere verfügbare eingebaute Panels.

The `EMBED_TAB_NAME` entry will used as the title for the new tab.(must be unique)
Tab `EMBED_TAB_LOCATION` options include: `tabWidget_utilities`, `tabWidget_setup` and `stackedWidget_mainTab` and `WINDOW`.
Tab `EMBED_TAB_COMMAND` specifies what embed-able program to run, including any of its command line options.

If using the `tabWidget_utilities` or `tabWidget_setup` locations, an extra tab will appear with the panel.
If using `stackedWidget_mainTab`, a button labelled *User* will appear.
Pressing this button will cycle through displaying all available panels (specified for this location) on the main tab area.
You also use `WINDOW` to create a pop up window that cab be shown/hidden with an arrow button that will appear near the machine on button

Einbetten einer Vismach Fräse

Sample zufügen ein builtin Diskussionsrunde zu den Nutzen Deckel, #d.h., ein graphisches #animieren Maschine benutzend die vismach Bibliothek.

```
[DISPLAY]
EMBED_TAB_NAME = Vismach Demo
EMBED_TAB_COMMAND = qtvcp vismach_mill_xyz
```

```
EMBED_TAB_LOCATION = tabWidget_utilities
```

Einbettung des Spindelriemen Panels

Dieses Beispielpanel dient zur Anzeige zusätzlicher RS485-VFD-Daten und zur Konfiguration eines 4-Scheiben-, 2-Riemen-Spindelantriebs über eine Reihe von Tasten (engl. buttons).




```
[DISPLAY]
EMBED_TAB_NAME = Spindle Belts
EMBED_TAB_COMMAND = qtvcp spindle_belts
EMBED_TAB_LOCATION = tabWidget_utilities
```

Pfade der Unterroutinen

Wenn Sie NGCGUI, Remap- oder benutzerdefinierte M-Code-Routinen verwenden, muss LinuxCNC wissen, wo die Dateien zu finden sind.
Dieses Beispiel ist typisch für das, was für NgcGui, Basic Probe und Versa Probe Remap-Code benötigt wird.
Diese Pfade müssen angepasst werden, um auf die tatsächlichen Dateien auf Ihrem System zu zeigen. RS274NZGC Abschnitt Details

```
[RS274NGC]
SUBROUTINE_PATH =
:~/linuxcnc/nc_files/examples/ngcgui_lib:~/linuxcnc/nc_files/examples/ngcgui_lib/utilitys
ubs: \
~/linuxcnc/nc_files/examples/probe/basic_probe/macros:~/linuxcnc/nc_files/examples/remap-
subroutines: \
~/linuxcnc/nc_files/examples/ngcgui_lib/remap_lib
```

QtVCPs NGCGUI-Programm von QtVCP muss auch wissen, wo es für die Auswahl und Vorauswahl von Unterprogrammen geöffnet werden muss.
NGCGUI_SUBFILE_PATH muss auf einen tatsächlichen Pfad auf Ihrem System zeigen und auch auf einen Pfad, der in SUBROUTINE_PATHS beschrieben ist.

```
[DISPLAY]
```

```
# NGCGUI Unterprogramm Pfad.
# Thr Pfad muss auch in [RS274NGC] SUBROUTINE_PATH sein
NGCGUI_SUBFILE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib
# vorgewählte Programme Tabs
# nur Dateinamen angeben, Dateien müssen sich im NGCGUI_SUBFILE_PATH befinden
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

Vorschau Kontrolle

Magische Kommentare können verwendet werden, um die G-Code-Vorschau zu steuern.

Bei sehr großen Programmen kann die Vorschau sehr lange zum Laden brauchen. Sie können steuern, was auf dem Grafikbildschirm angezeigt und was ausgeblendet wird, indem Sie die entsprechenden Kommentare aus dieser Liste in Ihren G-Code einfügen:

```
(PREVIEW,stop)
(PREVIEW,hide)
(PREVIEW,show)
```

Programmerweiterungen/Filter

Mit Programmerweiterungen können Sie steuern, welche Programme im Dateimanager-Fenster angezeigt werden.
Erstellen Sie eine Zeile mit den gewünschten .-Endungen, getrennt durch Kommata, dann ein Leerzeichen und die Beschreibung.
Sie können mehrere Zeilen für verschiedene Auswahlen im Kombinationsfeld hinzufügen.

```
[FILTER]
PROGRAM_EXTENSION = .ngc,.nc,.tap G-Code Datei (*.ngc,*.nc,*.tap)
```

QtDragon hat die Möglichkeit, geladene Dateien durch ein "Filterprogramm" zu schicken. Dieser Filter kann jede gewünschte Aufgabe erfüllen: Etwas so Einfaches wie sicherzustellen, dass die Datei mit "M2" endet, oder etwas so Kompliziertes wie die Erzeugung von G-Code aus einem Bild. Siehe Filterprogramme für weitere Informationen.

Der Abschnitt [FILTER] der INI-Datei steuert, wie die Filter funktionieren. Schreiben Sie zunächst für jeden Dateityp eine *PROGRAM_EXTENSION*-Zeile. Dann geben Sie das Programm an, das für jeden Dateityp ausgeführt werden soll. Dieses Programm erhält den Namen der Eingabedatei als erstes Argument und muss rs274ngc-Code in die Standardausgabe schreiben. Diese Ausgabe ist das, was im Textbereich angezeigt wird, in der Vorschau im Anzeigebereich, und dann auch von LinuxCNC ausgeführt wird.

Die folgenden Zeilen unterstützen den `image-to-gcode`-Konverter, der mit LinuxCNC und mit Python basierenden Filterprogrammen enthalten ist:

```
[FILTER]
PROGRAM_EXTENSION = .png,.gif,.jpg Greyscale Depth Image
PROGRAM_EXTENSION = .py Python Script
```

```
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
py = python
```

Sonden-/Touchplate-/Lasereinstellungen

QtDragon hält INI-Einträge für zwei optionale Sondierungs-Tab-Screens bereit. Kommentieren/unkommentieren Sie, was immer Sie bevorzugen.

- *Versa-Probe* ist eine auf QtVCP portierte Version des beliebten GladeVCP-Sondierungspanels.
- *Basic Probe* ist eine auf QtVCP portierte Version, die auf dem Basic Probe Screen eines Drittanbieters basiert.

Beide führen ähnliche Antastroutinen durch, wobei der Versa-Taster optional auch die automatische Werkzeugmessung übernimmt.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

Abbruch-Erkennung

Bei der Verwendung der Sondierungsroutinen von qtdragon ist es wichtig, eine Abbruchanfrage des Benutzers zu erkennen.
Standardmäßig meldet LinuxCNC einen Abbruch nicht in einer nützlichen Weise für die Probe-Routinen. Sie müssen eine ngc-Datei hinzufügen, um einen Fehler auszugeben, der erkannt werden kann. Details zum Abbruch neu zuordnen

```
[RS274NGC]
# bei abbruch wird diese ngc datei aufgerufen. erforderlich für basic/versa probe
Routinen. +
ON_ABORT_COMMAND=O <on_abort> call
```

Dieser Beispielcode sendet eine Nachricht bei Abbruch. Die Prüfroutinen können dieses Beispiel erkennen.
Gemäß der obigen Einstellung müsste es als "on_abort.ngc" in den [RS274NGC] SUBROUTINE_PATHS und [DISPLAY] PROGRAM_PREFIX Suchpfaden von LinuxCNC gespeichert werden.

```
o<on_abort> sub

o100 if [#1 eq 5]
    (machine on)
o100 elseif [#1 eq 6]
    (machine off)
o100 elseif [#1 eq 7]
    (estopped)
o100 elseif [#1 eq 8]
    (msg,Process Aborted)
```

```
o100 else
    (DEBUG,Abort Parameter is %d[#1])
o100 endif

o<on_abort> endsub
m2
```

Codes für die Inbetriebnahme

Sie sollten einen Standard-M/G-Code für den Start festlegen. Diese werden durch die Ausführung einer NGC-Datei außer Kraft gesetzt.
Dies sind nur Beispielcodes, der Integrator sollte geeignete Codes wählen.

```
[RS274NGC]
# G/M-Codes beim ersten Laden starten
RS274NGC_STARTUP_CODE = G17 G20 G40 G43H0 G54 G64P0.0005 G80 G90 G94 G97 M5 M9
```

MDI/Macro Buttons

QtDragon has up to ten convenience buttons for calling *MDI actions* or *macro actions*.
MDI actions are defined under the heading *[MDI_COMMAND_LIST]* as *MDI_COMMAND_MACRO0* = to *MDI_COMMAND_MACRO9* =
These could also call OWord routines if desired.
In the sample configurations they are labelled for moving between current user system origin (zero point) and Machine system origin.
User origin is the first MDI command in the INI list, machine origin is the second.
This example shows how to move Z axis up first. Commands separated by the *;* are run one after another.
The button label text can be set with any text after a comma, the \n symbol adds a line break.
The buttons can require the mode to be preset to MDI or to automatically switch from MANUAL to MDI mode by setting preferences on the settings page.
These commands can be run with external HAL pins, if using the hal_bridge component. See HAL_BRIDGE

```
[MDI_COMMAND_LIST]
# für Makro Buttons
MDI_COMMAND_MACRO0 = G0 Z25;X0 Y0;Z0, Goto\nUser\nZero
MDI_COMMAND_MACRO1 = G53 G0 Z0;G53 G0 X0 Y0,Goto\nMachn\nZero
```

It is also possible to call Oword programs that require user input at run time.
There are only 10 buttons available split between MDI_COMMAND and MACROS.
These are defined in the INI under [MACROS]

Here we show a sample that defines the call of the Oword program *increment*, which will prompt the user for two values *xinc* and *yinc* and will set the button text to *INCR*.
The second entry defines a call to the Oword file *lost* and will set the button text to *Lost*.

```
[MACROS]
```

```
MACRO_COMMAND_MACRO2 = increment xinc yinc ,INCR
MACRO_COMMAND_MACRO3 = lost ,Lost
```

NOTE All defined Oword files must be located on the system as defined in the [DISPLAY]PROGRAM_PREFIX or [RS274NGC]SUBROUTINE_PATH INI entries. Qtdragon will do checks for the file path when asked to run the Oword macro.

Post-GUI HAL-Datei

Diese optionalen HAL-Dateien werden aufgerufen, nachdem QtDragon alles andere geladen hat.
Sie können mehrere Zeilen für mehrere Dateien hinzufügen. Jede Datei wird in der Reihenfolge aufgerufen, in der sie erscheint.
Der Aufruf von HAL-Dateien, nachdem QtDragon bereits geladen ist, stellt sicher, dass die HAL-Pins von QtDragon verfügbar sind.

Beispiel mit typischen Einträgen für die Spezifikation von HAL-Dateien, die nach dem Start des QtDragon gelesen werden sollen. Passen Sie diese Zeilen an die tatsächlichen Anforderungen an.

```
[HAL]
POSTGUI_HALFILE = qtdragon_hd_postgui.hal
POSTGUI_HALFILE = qtdragon_hd_debugging.hal
```

Post-GUI HAL-Befehl

Diese optionalen HAL-Befehle werden ausgeführt, nachdem QtDragon alles andere geladen hat.
Sie können mehrere Zeilen hinzufügen. Jede Zeile wird in der Reihenfolge aufgerufen, in der sie erscheint.
Jeder HAL-Befehl kann verwendet werden.

Beispiel mit typischen Dateien in der INI-Datei zum Laden von Modulen, nachdem die GUI verfügbar ist. Passen Sie diese an Ihre tatsächlichen Anforderungen an.

```
[HAL]
POSTGUI_HALCMD = loadusr qtvcp test_probe
POSTGUI_HALCMD = loadusr qtvcp test_led
POSTGUI_HALCMD = loadusr halmeter
```

HAL Bridge

Hal Bridge ähnelt HALUI - es hat HAL Pins, die mit QtDragon kommunizieren.
Diese Pins können mit HALUI genutzt werden, um ein benutzerfreundlicheres Kontrollfeld zu erstellen.

- Dies kann berichten/den Button zur Anzeige der ausgewählten Achse anpassen.
- Die Schnellauf Geschwindigkeit/Inkremente werden berichtet.
- There is a cycle start and pause pin - these call the code in QtDragon rather then the motion controller.
  This allows custom behaviour, such as spindle lift to work with external buttons.

- If there are macros defined in the INI (see: Macro Buttons), there will be pins available to initiate them.
  These macros can require the mode to be preset to MDI or to automatically switch from MANUAL to MDI mode by setting preferences on the settings page.
- clear/reload the display
- shutdown the screen
- ok/cancel of dialogs and notify (error) messages

Beispiel-Eintrag. Entferne -d zur Vermeidung des Debug-Outputs.

```
[HAL]
HALBRIDGE= hal_bridge -d
```

Typische verfügbare HAL Pins:

```
Component Pins:
Owner   Type  Dir                Value  Name
    29  bool  OUT                FALSE  bridge.axis-x-is-selected
    29  bool  IN                 FALSE  bridge.axis-x-select
    29  bool  OUT                FALSE  bridge.axis-y-is-selected
    29  bool  IN                 FALSE  bridge.axis-y-select
    29  bool  OUT                FALSE  bridge.axis-z-is-selected
    29  bool  IN                 FALSE  bridge.axis-z-select
    29  bool  IN                 FALSE  bridge.cancel-in
    29  bool  IN                 FALSE  bridge.cycle-pause-in
    29  bool  IN                 FALSE  bridge.cycle-start-in
    29  bool  IN                 FALSE  bridge.ini-macro-cmd-MACRO2
    29  bool  IN                 FALSE  bridge.ini-macro-cmd-MACRO3
    29  bool  IN                 FALSE  bridge.ini-mdi-cmd-MACRO0
    29  bool  IN                 FALSE  bridge.ini-mdi-cmd-MACRO1
    29  real  OUT                    0  bridge.jog-increment
    29  real  OUT                    0  bridge.jog-increment-angular
    29  real  OUT                 3000  bridge.jog-rate
    29  real  OUT                  360  bridge.jog-rate-angular
    29  real  OUT                    0  bridge.jog-rate-angular-in
    29  real  IN                     0  bridge.jog-rate-in
    29  sint  OUT                   -1  bridge.joint-selected
    29  bool  IN                 FALSE  bridge.ok-in
    29  bool  IN                 FALSE  bridge.reload-display-in
    29  bool  IN                 FALSE  bridge.shutdown-in
```

Integrierte Beispielkonfigurationen

Die Beispielkonfigurationen *sim/qtdragon/* oder `sim/qtdragon_hd` sind bereits so konfiguriert, dass sie QtDragon als Bildschirm verwenden. Es gibt mehrere Beispiele, um verschiedene Maschinenkonfigurationen zu demonstrieren.

10.5.3. Tastenbelegungen

QtDragon ist nicht in erster Linie für die Verwendung einer Tastatur zur Maschinensteuerung gedacht. Es fehlen viele Tastenkombinationen, die zum Beispiel AXIS hat - aber Sie können eine Maus oder einen

Touchscreen verwenden.
Der Einfachheit halber gibt es mehrere Tastenkombinationen zur Steuerung der Maschine.

```
F1 - Estop ein/aus
F2 - Maschine ein/aus
F12 - Stil-Editor
Home - Start aller Verbindungen der Maschine
Escape - Abbruch der Bewegung
Pause - Maschinenbewegung anhalten
```

10.5.4. Buttons

Schaltflächen, die ankreuzbar sind, ändern ihre Textfarbe, wenn sie angekreuzt werden. Dies wird durch das Stylesheet/Thema gesteuert

10.5.5. Virtuelle Tastatur

QtDragon enthält eine virtuelle Tastatur für die Verwendung mit Touchscreens. Um die Tastatur zu aktivieren, markieren Sie das Kontrollkästchen Virtuelle Tastatur verwenden auf der Seite Einstellungen. Wenn Sie auf ein beliebiges Eingabefeld klicken, wie z.B. Sondenparameter oder Tooltabelleneinträge, wird die Tastatur angezeigt. Um die Tastatur auszublenden, führen Sie einen der folgenden Schritte aus:

- press the *HIDE* button on the virtual keyboard.
- Klicken Sie auf den MAIN page Button
- in den AUTO-Modus wechseln

Es ist zu beachten, dass bei der Verwendung der virtuellen Tastatur das Jogging der Tastatur deaktiviert ist.

10.5.6. HAL-Pins

Diese Pins sind spezifisch für den QtDragon-Bildschirm.
Es gibt natürlich viele weitere HAL-Pins, die für LinuxCNC angeschlossen werden müssen, um zu funktionieren.

Wenn Sie eine Aufforderung zum manuellen Werkzeugwechsel benötigen, fügen Sie diese Zeilen in Ihre Postgui-Datei ein.
QtDragon emuliert die hal_manualtoolchange HAL Pins - laden Sie nicht die separate HAL Komponente *hal_manualtoolchange*.

```
net tool-change      hal_manualtoolchange.change   <=  iocontrol.0.tool-change
net tool-changed     hal_manualtoolchange.changed  <=  iocontrol.0.tool-changed
net tool-prep-number hal_manualtoolchange.number   <=  iocontrol.0.tool-prep-number
```

Auch wenn Sie keinen automatischen Werkzeugwechsler haben, sollten Sie sicherstellen, dass diese Pins in einer der HAL-Dateien angeschlossen sind:

```
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

Dieser Eingangspin sollte verbunden werden, um den Sondenstatus anzuzeigen.

```
qtdragon.led-probe
```

Diese Pins sind Eingänge für die Spindel-VFD-Anzeige.
Die Volt- und Ampere-Pins werden zur Berechnung der Spindelleistung verwendet.

```
qtdragon.spindle-modbus-connection
qtdragon.spindle-modbus-errors
qtdragon.spindle-amps
qtdragon.spindle-fault
qtdragon.spindle-volts
```

This bool pin is an output to the spindle control to pause it.
You would connect it to `spindle.0.inhibit`.

```
qtdragon.spindle-inhibit
```

Die Anzeige der Spindelgeschwindigkeit und die LED "Spindel auf Geschwindigkeit" (engl. spindle-at-speed) von QtDragon erfordern, dass `spindle.0.speed-in` mit dem Rückmeldesignal der Spindelgeschwindigkeit verbunden wird.
Als Rückmeldung kann ein Encoder oder VFD Feedback verwendet werden, solange das Feedback in Umdrehungen pro Sekunde (RPS) erfolgt.
Wenn keine Rückmeldung verfügbar ist, können Sie die Anzeige so einrichten, dass sie die angeforderte Geschwindigkeit zeigt, indem Sie die Pins wie folgt verbinden:

```
net spindle-speed-feedback spindle.0.speed-out-rps => spindle.0.speed-in
```

This bool output pin can be connected to turn on a laser:

```
qtdragon.btn-laser-on
```

This real output pin indicates the camera rotation in degrees:

```
qtdragon.cam-rotation
```

These bool/sint/real pins are related to external offsets if they are used:

```
qtdragon.eoffset-clear
qtdragon.eoffset-enable
qtdragon.eoffset-value
qtdragon.eoffset-spindle-count
qtdragon.eoffset-zlevel-count
qtdragon.eoffset-is-active
```

These real output pins reflect the current slider jograte (in machine units):

```
qtdragon.slider-jogspeed-linear
qtdragon.slider-jogspeed-angular
```

These real output pins reflect the current slider override rates:

```
qtdragon.slider-override-feed
qtdragon.slider-override-maxv
qtdragon.slider-override-rapid
qtdragon.slider-override-spindle
```

Diese Ausgangspins sind verfügbar, wenn Sie die Option Versa Probe INI einstellen. Sie können für auto-tool-length-probe bei Werkzeugwechsel verwendet werden - mit hinzugefügtem Remap-Code.

```
qtversaprobe.enable
qtversaprobe.blockheight
qtversaprobe.probeheight
qtversaprobe.probevel
qtversaprobe.searchvel
qtversaprobe.backoffdist
```

Dieser Pin wird wahr sein, wenn das geladene Werkzeug die in der Voreinstellungsdatei gesetzte Nummer in der Versa-Sonde-Toolnummer entspricht.
Es kann zum Beispiel verwendet werden, um die Spindel zu hemmen, wenn die Sonde durch die Verbindung mit `spindle.0.inhibit`.

```
qtversaprobe.probe-loaded
```

Dieser Ausgabe Pin ist verfügbar wenn die Basic Probe INI Option gesetzt ist.
Dieser Pin wird sein wahr sein, wenn das geladene Werkzeug der Nummer entspricht, die in der Basic Probe tool Nummer Edit Dialogbox gesetzt ist.
Der Pin kann benutzt werden (beispielsweise), um die Spindel zu sperren wenn die Sonde geladen ist, in dem er mit `spindle.0.inhibit` verbunden wird.

```
qtbasicprobe.probe-loaded
```

Dieser Eingabepin steht zur Verfügung, um Pause/Fortsetzen (engl. resume) eines laufenden Programms zu wechseln.

```
qtdragon.external-pause
```

Die meisten Qtdragon-Dialoge können extern über Dialogantworten gesteuert werden.

```
qtdragon.dialog-ok
qtdragon.dialog-no
qtdragon.dialog-cancel
```

10.5.7. HAL-Dateien

Die mitgelieferten HAL-Dateien sind nur für die Simulation gedacht. Eine reale Maschine benötigt ihre eigenen HAL-Dateien. Der QtDragon-Bildschirm funktioniert mit 3 oder 4 Achsen mit einem Gelenk pro Achse oder 3 oder 4 Achsen in einer Gantry-Konfiguration (2 Gelenke auf 1 Achse).

10.5.8. Manueller Werkzeugwechsel

Wenn Ihre Maschine einen manuellen Werkzeugwechsel erfordert, kann QtDragon ein Meldungsfenster anzeigen, um Sie anzuleiten. QtDragon emuliert die hal_manualtoolchange HAL Pins - laden Sie nicht die separate HAL Komponente *hal_manualtoolchange*. Dazu müssen Sie z.B. den richtigen HAL-Pin in der postgui HAL-Datei verbinden:

```
net tool-change      hal_manualtoolchange.change   <=  iocontrol.0.tool-change
net tool-changed     hal_manualtoolchange.changed  <=  iocontrol.0.tool-changed
net tool-prep-number hal_manualtoolchange.number   <=  iocontrol.0.tool-prep-number
```

10.5.9. Spindel

Der Bildschirm ist für den Anschluss an einen VFD gedacht, funktioniert aber auch ohne ihn.
Es gibt eine Reihe von VFD-Treiber in der LinuxCNC Distribution enthalten.
Es ist bis zu dem Endbenutzer, um die entsprechenden Treiber und HAL-Datei-Verbindungen nach seiner eigenen Maschine Setup liefern.

10.5.10. Automatisches Anheben der Z-Achse bei Programm-Pause

QtDragon kann eingerichtet werden, um die Z-Achse automatisch zu erhöhen und zu senken und die Spindel zu stoppen, wenn das Programm angehalten wird (engl. paused).
Sie schalten die Taste *SPINDLE LIFT* oder *NO LIFT* im Wechsel um die Option die Spindel in Z zu erhöhen, wenn angehalten wurde.
Wenn Sie dann die Taste "PAUSE" drücken, wird die Spindel den Betrag auf der Registerkarte "Einstellungen" angehoben und die Spindel wird anhalten.
Das Drücken von *RESUME* wird die Spindel wieder starten und senkt die Spindel.
Wenn Sie den HAL-Pin `spindle.0.at-speed` mit einem Treiber-Pin verbunden haben, wird die Spindel erst dann abgesenkt, wenn der Pin wahr ist
Normalerweise verbinden Sie diesen Pin mit einem Timer oder einer Logik, welche die Geschwindigkeit der Spindel erkennt.
Wenn dieser Pin nicht mit einem Antriebsstift verbunden ist, wird ein Dialogfeld angezeigt, in dem Sie darauf hingewiesen werden, dass Sie auf ein Erreichen der Geschwindigkeit der Spindel warten müssen.
Die Spindel wird abgesenkt, wenn Sie dieses Dialogfeld schließen.
Der Betrag, um den die Spindel angehoben werden soll, wird auf der Registerkarte "Einstellungen" unter der Überschrift "SPINDELERHÖHUNG" festgelegt.
Dieses Zeilen-Eingabefeld kann nur direkt eingestellt werden, wenn es sich nicht im Auto-Modus befindet.
Mit den Auf-/Ab-Tasten kann der Anhebungsbetrag jederzeit eingestellt werden, auch wenn die Spindel bereits angehoben ist.
Die Tastenschritte sind 1 Zoll oder 5 mm (je nach den Einheiten, auf denen die Maschine basiert).

NOTE Wenn Sie die Option Spindle Lift verwenden, kann HALUI nicht verwendet werden, um das Programm zu unterbrechen/abzunehmen. Es gibt einen Pin, `QtDragon.extern-pause` für ein Pausieren/Weitermachen von einer externen Quelle. Sie müssen auch externe Offsets aktivieren. In der Registerkarte "externe Offsets verwenden" hierzu anklicken. Wenn Sie die Spindel inhibieren möchten, wenn ein Sondenwerkzeug geladen wird, müssen Sie eine logische `or`-Komponente (engl. für oder) verwenden, um die beiden Spindelhemmsignale zu kombinieren und mit um mit Spindel `spindle.0.inhibit` zu verbinden.

Dieses optionale Verhalten erfordert Ergänzungen zur INI und der QtDragon_postgui HAL-Datei.

In der INI, unter der Überschrift AXIS_Z.

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.2
```

Dies behält sich 20% der maximalen Geschwindigkeit und maximale Beschleunigung für die externen Versätze vor.
Dies begrenzt die maximale Geschwindigkeit der Maschine um 20%

In der Datei qtdragon_postgui.hal hinzufügen:

```
# Externe Offsets der Z-Achse einrichten
net eoffset_clear    qtdragon.eoffset-clear          =>  axis.z.eoffset-clear
net eoffset_count    qtdragon.eoffset-spindle-count  =>  axis.z.eoffset-counts
net eoffset          qtdragon.eoffset-value          <=  axis.z.eoffset
net eoffset-state    qtdragon.eoffset-is-active      <=  motion.eoffset-active

# Inhibieren der Spindel bei Unterbrechung (engl. pause)
net spindle-pause    qtdragon.spindle-inhibit        =>  spindle.0.inhibit

# uncomment for dragon_hd
#net limited          qtdragon.led-limits-tripped     <= motion.eoffset-limited

setp axis.z.eoffset-enable 1
setp axis.z.eoffset-scale 1.0
```

10.5.11. Z-Level-Kompensation

QtDragon_hd kann mit dem externen Programm *G-code Ripper* so eingestellt werden, dass es Höhenänderungen der Z-Ebene prüft und ausgleicht.

NOTE Diese Funktion ist nur in der Version QtDragon_hd verfügbar.

Z-Ebene Ausgleich ist ein Bett Nivellierung / Verzerrung Korrekturfunktion typischerweise in 3D-Druck oder Gravur verwendet. Es verwendet eine HAL Nicht-Echtzeit-Komponente, welche die externen Offsets Funktion von LinuxCNC verwendet. Die Komponente hat einen HAL-Pin, der einen Interpolationstyp spezifiziert, der entweder kubisch, linear oder am nächsten (0, 1, 2) sein muss. Wenn keine angegeben ist oder wenn eine ungültige Zahl angegeben ist, wird die Standardeinstellung kubisch

sein.

Wenn Z LEVEL COMP aktiviert ist, liest die Kompensationskomponente eine Sondendaten-Datei, die *probe_points.txt* heißen muss. Die Datei kann jederzeit geändert oder aktualisiert werden, solange die Kompensation deaktiviert ist. Bei der nächsten Aktivierung wird die Datei erneut gelesen und die Kompensationskarte wird neu berechnet. Diese Datei sollte sich im Konfigurationsverzeichnis befinden.

Die Sondierungsdatei wird von einem Sondierungsprogramm erzeugt, das seinerseits von einem externen Python-Programm namens `gcode_ripper` erzeugt wird, das auf der Registerkarte "Dateimanager" über die Schaltfläche "G-code Ripper" aufgerufen werden kann.

Verwendung von G-code Ripper für die Z-Ebenen-Kompensation

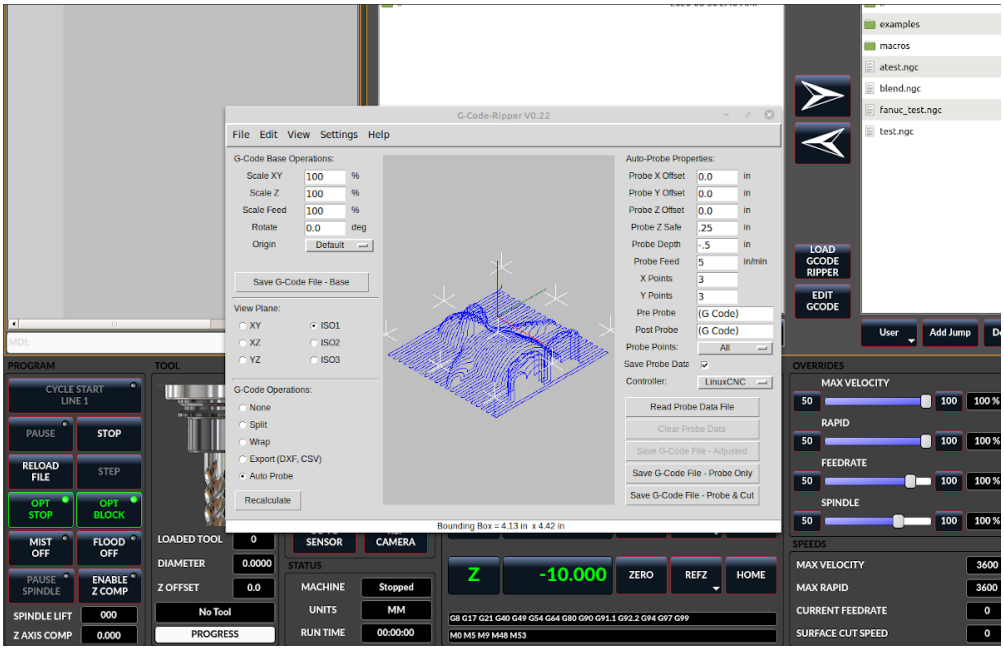


Figure 177. QtDragon_hd zeigt G-code Ripper

NOTE G-code Ripper bietet viele Funktionen, auf die wir hier nicht näher eingehen werden. Diese sind nur in der QtDragon_hd Version verfügbar.

- Wechseln Sie in qtdragon_hd auf die Registerkarte Datei und drücken Sie die Schaltfläche G-Code Ripper laden.
- Ursprung so einstellen, dass er mit dem Ursprung der zu prüfenden G-Code-Datei übereinstimmt.
- Aktivieren Sie unter G-Code-Operationen die Option Auto Probe.
- Datei → G-Code Datei öffnen (Die Datei, die Sie nach der Kompensation ausführen)
- Falls erforderlich, nehmen Sie Anpassungen vor und drücken Sie Neu berechnen.
- Drücken Sie G-Code-Datei speichern - nur Sonde.
- Speichern Sie die erzeugte Datei im Ordner nc_files.
- Beenden Sie gcode_ripper.
- Im Ordner nc_files sollte sich nun eine Datei mit dem Namen {something}-probe-only.ngc befinden. Setzen Sie den Dateifilter auf G-Code-Dateien, navigieren Sie zum Verzeichnis nc_files und laden Sie

diese Datei.

- Führen Sie dieses Programm aus, ohne die Offsets zu ändern. Stellen Sie sicher, dass das Sondenwerkzeug installiert ist. Nach Abschluss des Programms befindet sich im Konfigurationsverzeichnis eine Datei mit dem Namen "probe_points.txt".
- Drücken Sie in qtdragon_hd die Schaltfläche "Enable Z Comp", um den Ausgleich zu aktivieren. Schauen Sie in der Statuszeile nach, ob die Kompensation erfolgreich war oder nicht. Die aktive Kompensation wird neben der Beschriftung "Z Level Comp" angezeigt. Während des Joggens sollte sich diese Anzeige je nach Kompensationskomponente ändern.

NOTE Wenn Sie die automatische Anhebung Z verwenden, um die Spindel in der Pause anzuheben, müssen Sie die beiden mit einer HAL-Komponente kombinieren und diese an die Bewegungs (motion)-Komponente von LinuxCNC weiterleiten.

Beispiel einer postgui HAL-Datei für kombinierte Spindelanhebung und Z-Niveau-Kompensation

```
# Komponenten laden
#############################################################################

loadrt logic names=logic-and personality=0x102
addf logic-and servo-thread

# load a summing component for adding spindle lift and Z compensation
loadrt scaled_sint_sums
addf scaled-sint-sums.0 servo-thread

loadusr -Wn z_level_compensation z_level_compensation
# method parameter must be one of nearest(2), linear(1), cubic (0)
setp z_level_compensation.fade-height 0.0
setp z_level_compensation.method 1

# Signale mit der Bewegungskomponente von LinuxCNC verbinden
#############################################################################

net eoffset-clear    axis.z.eoffset-clear
net eoffset-counts   axis.z.eoffset-counts
setp axis.z.eoffset-scale .001
net eoffset-total          axis.z.eoffset
setp axis.z.eoffset-enable True

# Externe Offsets für die Spindelpausenfunktion
#############################################################################
net eoffset-clear             qtdragon.eoffset-clear
net eoffset-spindle-count   <= qtdragon.eoffset-spindle-count
net spindle-pause             qtdragon.spindle-inhibit        =>  spindle.0.inhibit
net eoffset-state             qtdragon.eoffset-is-active      <=  motion.eoffset-active

## Z level Kompensation
#####################################################
net eoffset-clr2           z_level_compensation.clear      => logic-and.in-01
net xpos-cmd               z_level_compensation.x-pos      <= axis.x.pos-cmd
net ypos-cmd               z_level_compensation.y-pos      <= axis.y.pos-cmd
net zpos-cmd               z_level_compensation.z-pos      <= axis.z.pos-cmd
net z_compensation_on      z_level_compensation.enable-in  <= qtdragon.comp-on
```

```
net eoffset-zlevel-count    z_level_compensation.counts     => qtdragon.eoffset-zlevel-
count

# add Z level and scaled spindle raise level values together
net eoffset-spindle-count   scaled-sint-sums.0.in0
net eoffset-zlevel-count    scaled-sint-sums.0.in1
setp scaled-sint-sums.0.scale0 1000
net eoffset-counts          scaled-sint-sums.0.out-s
```

10.5.12. Sondieren

Der Sondenbildschirm wurde einem grundlegenden Test unterzogen, könnte aber noch einige kleinere Fehler aufweisen. Gehen Sie bei der Ausführung von Prüfroutinen äußerst vorsichtig vor, bis Sie mit der Funktionsweise vertraut sind. Die Prüfroutinen laufen, ohne die Haupt-GUI zu blockieren. Dies gibt dem Bediener die Möglichkeit, die DROs zu beobachten und die Routine jederzeit zu stoppen.

NOTE Die Sondierung ist sehr unempfindlich gegenüber Fehlern; überprüfen Sie die Einstellungen vor der Verwendung.

QtDragon verfügt über 2 Methoden zur Einstellung von Z0. Die erste ist eine Tastplatte, bei der eine Metallplatte mit bekannter Dicke auf das Werkstück gelegt wird, dann wird das Werkzeug abgesenkt, bis es die Platte berührt, wodurch das Messtastersignal ausgelöst wird. Der Z0-Wert des aktuellen Anwendersystems (G5x) wird auf die Messtasterhöhe - die eingegebene Blechdicke - eingestellt.

Bei der zweiten Methode wird ein Werkzeugeinstellgerät in einer festen Position und einer bekannten Höhe über dem Tisch verwendet, wo das Messtastersignal ausgelöst wird. Um Z0 auf die Oberseite des Werkstücks einzustellen, muss bekannt sein

1. wie weit der Auslösepunkt des Messtasters über dem Tisch liegt (Höhe der Werkzeugaufnahme) und
2. wie weit über dem Tisch sich die Werkstückoberseite befindet.

Dieser Vorgang muss bei jedem Werkzeugwechsel durchgeführt werden, da die Werkzeuglänge nicht gespeichert wird.

Beim Antasten mit einem Tastsystem wird der Parameter Höhe vom Tisch bis zur Oberkante des Werkstücks nicht berücksichtigt und kann ignoriert werden, unabhängig davon, ob Sie den Tastplattenbetrieb mit einer auf 0 eingestellten Dicke oder eine Antastroutine verwenden. Er gilt nur für den Werkzeugeinrichter.

Versa Taster (engl. probe)

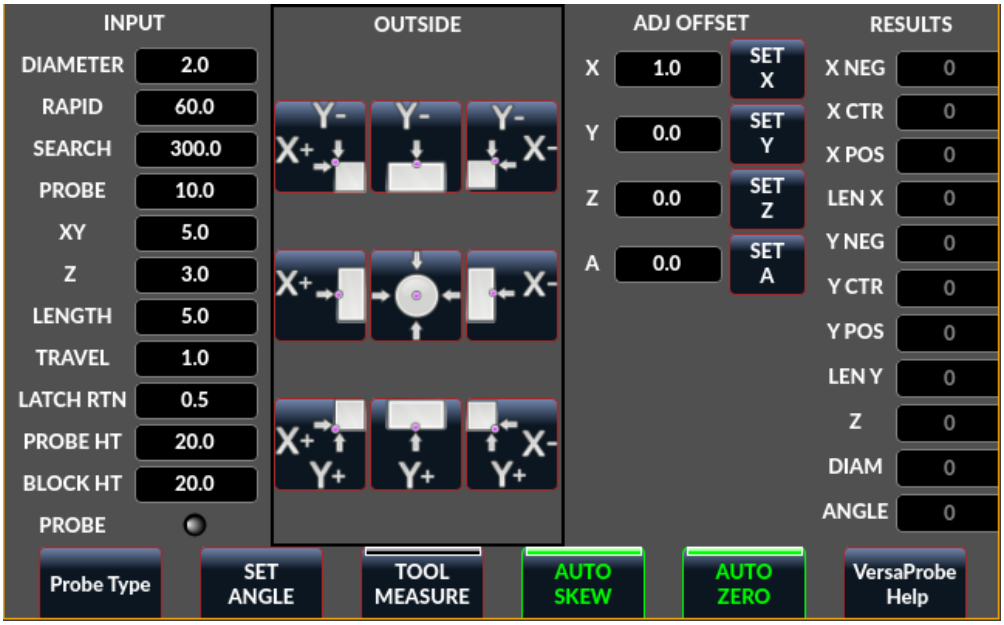


Figure 178. QtDragon - Versa-Probe-Option

Der Versa-Taster wird zum halbautomatischen Antasten von Werkstücken verwendet, um Kanten, Mittelpunkte und Winkel zu finden.
Er kann auch zum automatischen Antasten der Werkzeuglänge beim Werkzeugwechsel mit zusätzlichem Remap-Code verwendet werden.

Sie müssen die "Sondierungs-Parameter" sorgfältig einstellen:

DIAMETER

Dies ist der Durchmesser der Sondenspitze. Die Genauigkeit der Sondenmessungen wird direkt von der Genauigkeit des Durchmessers der Sondenspitze beeinflusst.

TRAVEL

Minimaler zurückgelegte Entfernung der Sonde bei der ersten Suche. Wenn die Suchstrecke zu kurz ist, erhalten Sie eine Meldung wie "G38 beendet ohne Kontakt". Aus Sicherheitsgründen wird empfohlen, diesen Parameter auf 3-4 mm mehr als den Durchmesser des Tastereinsatzes einzustellen.

LATCH RTN

Der Abstand, um den die Sonde nach dem ersten Kontakt mit dem Werkstück zurückgezogen wird. Dieser Abstand sollte kurz sein, da die zweite Annäherung mit einer langsamen Geschwindigkeit erfolgt, aber groß genug, damit der Messtaster den Kontakt unterbricht und in den Zustand der Suchbereitschaft gebracht wird. Wenn der Abstand Latch Rtn zu groß ist, verbringen Sie viel Zeit damit, auf den Abschluss der Suche zu warten. Empfehlung: 1-2 mm

SEARCH

Dies ist die Vorschubgeschwindigkeit des Messtaster während der Suche nach dem Ziel, in Maschineneinheiten pro Minute. Die Suchgeschwindigkeit sollte langsam genug sein, um eine akzeptable Anfangsgenauigkeit zu erreichen, aber schnell genug, um keine Zeit mit dem Warten auf eine Bewegung zu verlieren. Empfehlung: 200-500 mm/min.

***PROBE* (engl. für Taster oder Sensor)**

Sobald der erste Kontakt hergestellt und der Taster zurückgezogen ist, wartet er 0,5 Sekunden, bevor er die Suche mit einer niedrigeren Geschwindigkeit, der Tastergeschwindigkeit, erneut durchführt. Diese niedrigere Geschwindigkeit stellt sicher, dass die Maschine die Bewegung bei Kontakt mit dem Werkstück so schnell wie möglich stoppen kann.

RAPID

Achsbewegungen, die nicht mit der Suche verbunden sind, werden mit der durch RAPID definierten Geschwindigkeit in Maschineneinheiten pro Minute ausgeführt.

SIDE/EDGE LENGTH

Dies ist die Entfernung, die der Messtaster im Eiltempo zu der Position fährt, an der er mit der Suche beginnt. Wenn eine Ecke gemessen wird, bewegt sich der Messtaster EDGE LENGTH von der Ecke weg, bewegt sich dann um XY CLEARANCE vom Werkstück weg, senkt sich um Z CLEARANCE und beginnt mit der Anfangssuche. Wenn Sie einen inneren Kreis messen, sollte EDGE LENGTH auf den ungefähren Radius des Kreises eingestellt werden. Hinweis: NICHT der Durchmesser.

PROBE HT

Die Höhe des Werkzeugsensors von der Oberfläche des Maschinentisches. Dieser Wert wird verwendet, um die Z-Null-Höhe für das aktuelle Arbeitskoordinatensystem zu berechnen, wenn der Messtaster mit einem Werkzeugeinstellsensor verwendet wird.

BLOCK HT

Die Höhe der Oberseite des Werkstücks von der Maschinentischoberfläche. Dieser Wert wird verwendet, um die Z-Nullhöhe für das aktuelle Werkstückkoordinatensystem zu berechnen, wenn der Messtaster mit einem Einrichtungssensor verwendet wird.

XY CLEARANCE

Der Abstand, den der Messtaster von einer Kante oder Ecke wegfährt, bevor er eine Suche durchführt. Er sollte groß genug sein, um sicherzustellen, dass der Messtaster weder das Werkstück noch andere Vorrichtungen berührt, bevor er nach unten fährt. Er sollte klein genug sein, um übermäßiges Warten auf die Bewegung während der Suche zu vermeiden.

Z CLEARANCE

Der Abstand, den die Sonde nach unten fährt, bevor sie eine Suche durchführt. Wenn Sie eine Innenbohrung messen, können Sie den Taster manuell auf die Z-Starthöhe verfahren und dann Z CLEARANCE auf 0 setzen.

Dort sind drei Umschalttasten (engl. toggle buttons):

***Auto Zero* (Auto-Nullstellung)**

Hiermit wird bestimmt, ob die betreffende Achse nach der Abtastung im aktuellen Anwendersystem auf Null gesetzt wird.

Auto Skew

Hiermit wird festgelegt, ob das System nach dem Antasten gedreht wird oder nur die berechnete Drehung angezeigt wird.

***Tool Measure* (Werkzeugmessung)**

Diese Option (falls integriert) schaltet die automatische Werkzeugmessung ein und aus.

HAL-Pins

Versaprobe bietet 5 Ausgabepins für die Werkzeugmessung und einen um die Spindel zu unterbrechen während die Sonde geladen wird.
Die Pins werden von einem Remap-G-Code-Unterprogramm gelesen, so dass der Code auf verschiedene Werte reagieren kann.
Derzeit kann die Sonden-Werkzeugnummer (engl. probe tool number) nur in der Einstellungs-Datei verändert werden:

```
[VERSA_PROBE_OPTIONS]
ps_probe_tool = 1
```

`qtversaprobe.enable` (HAL_BOOL)

Messung aktiviert oder nicht Werkzeug. Spiegelt den Zustand des Bildschirmtaste (engl. screen button) wider.

`qtversaprobe.blockheight` (HAL_REAL)

Die gemessene Höhe der Oberseite des Werkstücks. Spiegelt die Eingabe am Bildschirm wider.

`qtversaprobe.probeheight` (HAL_REAL)

Die Höhe des Schalters für die Werkzeugsonde. Spiegelt die Eingabe am Bildschirm wider.

`qtversaprobe.searchvel` (HAL_REAL)

Die Geschwindigkeit, mit der nach dem Schalter für den Werkzeugmesstaster gesucht wird

`qtversaprobe.probevel` (HAL_REAL)

Die Länge des Werkzeugs zwischen Geschwindigkeit und Messtaster. Spiegelt die Eingabe am Bildschirm wider.

`qtversaprobe.backoffdist` (HAL_REAL)

Die Entfernung, um die sich die Sonde nach dem Auslösen zurückzieht. Entsprich der Bildschirmeingabe.

`qtversaprobe.probe-loaded` (HAL_BOOL)

Reflektiert, ob das aktuelle Werkzeug der in der Voreinstellungsdatei (engl. preferences)-Datei eingestellten Sondennummer entspricht.

Einfache Sonde

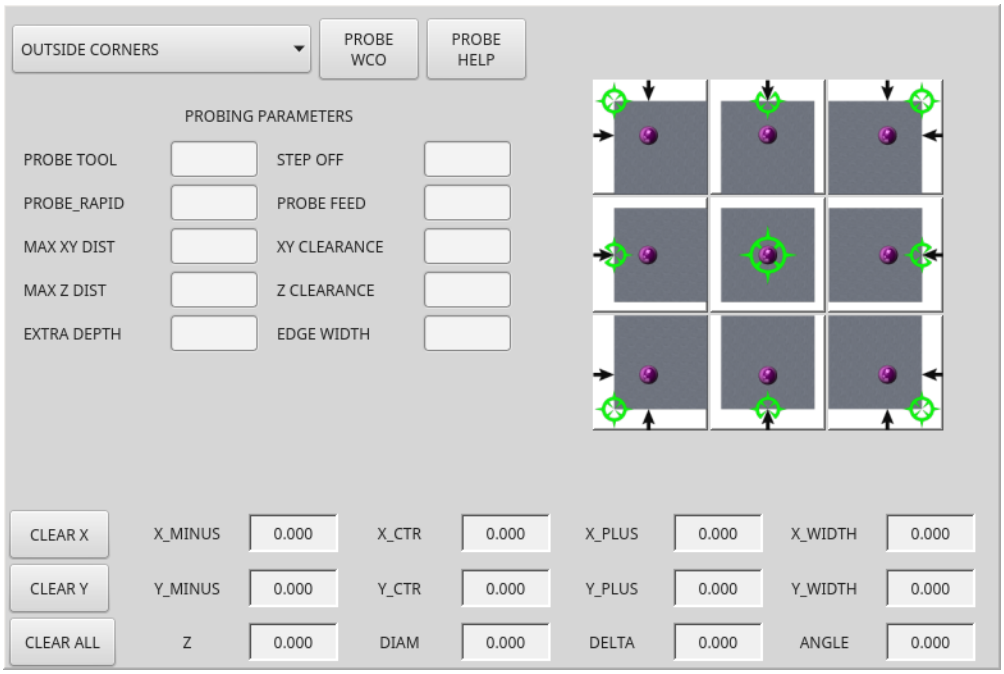


Figure 179. QtDragon - Grundlegende Sondenoption

Der Basistaster wird zum halbautomatischen Antasten von Werkstücken verwendet, um Kanten, Mittelpunkte und Winkel zu finden. Das Kombinationsfeld ermöglicht die Auswahl des Grundtyps der gezeigten Messtasten:

- Außenecken
- Innenecken
- Kantenwinkel
- Erhebungen und Taschen (engl. boss and pockets)
- Erhöhungen und Vertiefungen
- Kalibrierung

Sie müssen die "Sondierungs-Parameter" sorgfältig einstellen:

***Probe Tool* (engl. buchstäblich Sonden-Werkzeug)**

Erlaubt das Abtasten nur, wenn sich diese Werkzeugnummer in der Spindel befindet

***Probe Diameter* (Sonden-Durchmesser)**

Die Größe der Sonden-Spitze

Probe Rapid

Die Geschwindigkeit schneller Bewegungen der Sonde in Maschineneinheiten

Probe Search

Die Geschwindigkeit des ersten *groben* Abtastens in Maschineneinheiten

Sonden-Vorschub

Die Geschwindigkeit der zweiten "Feinsuche" in Maschineneinheiten

Step Off

Zurückziehen und erneutes Abtasten der Entfernung

Max XY Distance

Die maximale Entfernung, in der die Sonde in X und Y sucht, bevor sie mit einem Fehler abbricht

Max Z Distance

Die maximale Entfernung, in der die Sonde in Z sucht, bevor sie mit einem Fehler abbricht

XY Clearance

Abstand zwischen Sonde und Wandkante vor dem schnellen Abwärtsfahren in Z und dem "groben" Antasten

Z Clearance

Abstand zwischen Sonde und Oberkante des Materials

Extra Depth

Abstand von der Oberkante des Materials bis zur gewünschten Sondentiefe

Es gibt auch Hinweisparameter, abhängig von der gewählten Tastart:

Edge Width

Gewünschter Abstand von der Startposition der Sonde entlang der Wandkante, bevor die Sonde gestartet wird

Diameter Hint

(engl. wörtlich Durchmesser-Hinweis) Wird beim Rundkopf- oder Rundtaschen-Sondieren verwendet (Startbewegung: 1/2 Durchmesser plus XY-Abstand)

X Hint

(wörtlich engl. X-Hinweis) wird von rechteckigen Vorsprüngen/Taschen verwendet (Startbewegung: 1/2 X-Länge plus XY-Abstand)

Y Hint

Wird von der rechteckigen Aufsatz-/Taschenmessung verwendet (Startbewegung: 1/2 Y-Länge plus XY-Abstand)

Nach der Einstellung der Parameter und Berücksichtigen der Hinweise:

- Bewegen Sie die Sonde manuell zu der ungefähren Position, die durch das grüne Ziel auf der Schaltfläche dargestellt wird.
- Bestätigen Sie, dass die Parameter angemessen sind.
- Drücken Sie die gewünschte Sondierungstaste.

Die Sondierungsroutine beginnt sofort.

NOTE Durch Drücken der Stopptaste oder der Escape-Taste der Tastatur wird die Sondierung

abgebrochen.

HAL-Pins

Dieser kann genutzt werden, die Spindel anzualten, wenn eine Sonde aufgenommen wird.
Sie verbinden ihn mit `spindle.0.inhibit`.

```
qtbasicprobe.probe-loaded
```

Beispiel für eine Ecksonde

Lassen Sie uns die Innenecken-Antastung mit der Schaltfläche oben rechts in Basic Probe (back_right_inside) besprechen. Während alle Sondeneinträge korrekt sein müssen, sind die wichtigsten Einstellungen für jede einzelne Sonde zu ändern:

XY CLEARANCE

Abstand von der Kante vor dem groben Antasten,

Z CLEARANCE

Abstand von der Sonde zur Oberseite des Materials,

EXTRA DEPTH

Abstand zum Absenken der Sonde von der Oberseite des Materials,

EDGE WIDTH

Abstand entlang der Kantenwand (von der Ecke weg) bis zum Beginn der Sondierung.

NOTE Dieser Abstand ist immer in "Maschineneinheiten" anzugeben (mm für metrische Maschine, Zoll für imperiale Maschinen).

Voreinstellung:

- Setzen Sie den Taster manuell auf den Schnittpunkt der Kanten (d.h. Ecke) des Materials, wie durch das grüne Bullauge auf der Taste beschrieben. Stellen Sie ihn Z CLEARANCE über der Oberseite des Materials ein. Diese Einstellungen können nach Augenmaß vorgenommen werden.
- Setzen Sie EXTRA CLEARANCE auf einen Wert, bei dem die Sonde unter die Oberkante des Materials fahren soll. (Die Sonde bewegt sich also von ihrer Startposition aus nach unten Z-Abstand + Extra-Abstand.)
- Setzen Sie XY CLEARANCE auf einen Wert, der einen gewissen Abstand zur Wand gewährleistet, damit die Sonde beim Absenken nicht gegen etwas stößt.
- setzen Sie EDGE WIDTH auf einen Wert, der den von der Ecke gemessenen Abstand entlang der Wand beschreibt, wo Sie Sonden wünschen. dieser Kantenabstand wird entlang der X-Wand und dann der Y-Wand verwendet.

Abfolge nach Betätigung der Sondentaste:

1. Rapid EDGE WIDTH (engl. für Schneller KANTENBREITE) Abstand von der Ecke sich entfernend, gleichzeitig XY CLEARANCE (engl. für XY-FREIRAUM) sich von der Kante weg bewegend. Dies ist also eine leicht diagonale Bewegung.
2. Sonde nach unten bewegen um Z CLEARANCE + EXTRA DEPTH,
3. Wand zweimal sondieren (grob (engl. rough) und fein),
4. diagonal zur anderen Wand bewegen, wie durch EDGE WIDTH und XY CLEARANCE festgelegt,
5. Wand zweimal sondieren,
6. Sonde nach oben bewegen um Z CLEARANCE + EXTRA DEPTH (kehrt zum Ausgangspunkt zurück,
7. Im Eilgang zurück zur startecke (jetzt berechnet anhand der mit der Sonde bestimmten Wänden),
8. Wenn die Schaltfläche für die automatische Nullung aktiviert ist, setzen von X und Y des aktuellen Benutzersystems auf Null.

Anpassen des Probe-Screen-Widgets

Es ist möglich, eine angepasste Version des Antast-Widgets zu laden.

Es sollte im Konfigurationsordner einen Ordner für Bildschirme geben: benannt <CONFIG FOLDER>/qtvcp/.
Darin kann (oder kann hinzugefügt werden) ein Ordner *lib/* und *widgets/* sein.
Im Ordner *widgets* können Sie *basic_probe.py* (oder *versa_probe.py*) und *probe_subprog.py* kopieren.
Im Ordner *lib* kopieren Sie *touchoff_subprogram.py*.
Wenn diese Dateien gefunden werden, werden sie anstelle der Originale verwendet.
Sie können die Dateien ändern, um das Verhalten anzupassen.

10.5.13. Touch-Platte

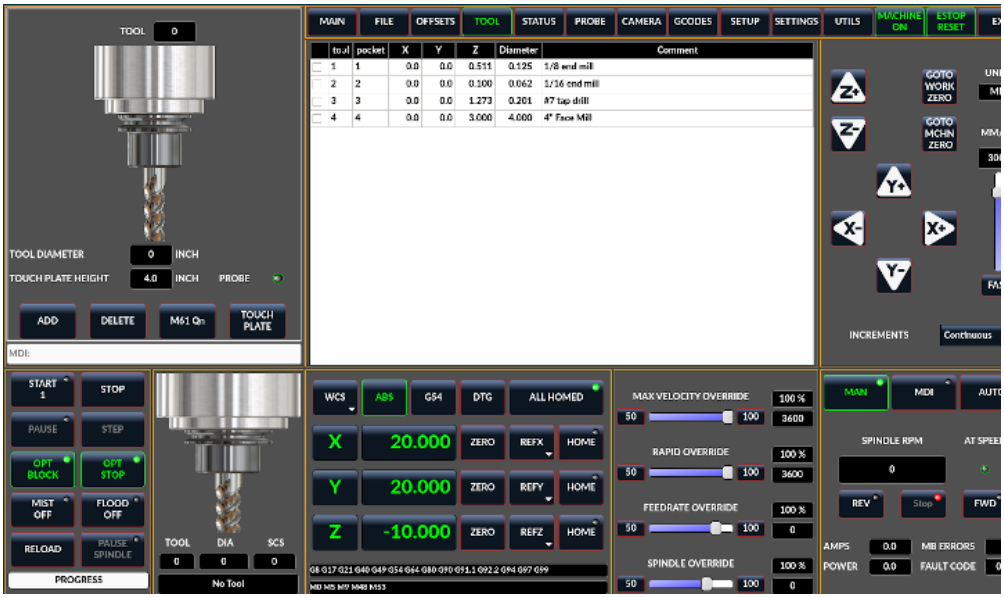


Figure 180. QtDragon Touch-Fläche (engl. touch plate)

Sie können eine leitfähige Tastplatte oder etwas Gleichwertiges verwenden, um die Z-Position eines Werkzeugs automatisch anzutasten (die Benutzerkoordinate auf Null zu setzen). Vor dem Antasten muss

ein Werkzeug geladen sein. Legen Sie auf der Registerkarte "Werkzeug" oder "Einstellungen" die Höhe der Tastplatte, die Such- und Antastgeschwindigkeit und den maximalen Antastabstand fest.

NOTE

Wenn Sie eine leitfähige Platte verwenden, sollten die Such- und Tastgeschwindigkeit gleich langsam sein. Wenn Sie einen Werkzeugwechsler mit gefedertem Verfahrweg verwenden, können Sie die Suchgeschwindigkeit schneller einstellen. LinuxCNC fährt die Geschwindigkeit mit der maximalen Beschleunigung herunter, so dass nach dem Auslösen des Messtasters ein Verfahrweg vorhanden sein kann, wenn die Geschwindigkeit zu hoch eingestellt ist.

Legen Sie die Platte auf die Oberfläche, auf der Sie Z nullen möchten. Verbinden Sie das Eingangskabel des Messtasters mit dem Werkzeug (bei Verwendung einer leitfähigen Platte). Bewegen Sie das Werkzeug manuell innerhalb des maximalen Tasterabstandes. Drücken Sie die Taste "Tastplatte". Die Maschine tastet zweimal nach unten und der aktuelle Benutzer-Offset (G5X) wird am unteren Ende der Platte durch Berechnung der Tastplattenhöhe auf Null gesetzt.

10.5.14. Automatische Werkzeugvermessung

Übersicht

QtDragon kann eingerichtet werden, um eine integrierte automatische Werkzeug-Messung mit dem Versa Sonde Widget und Remap-Code vorzunehmen.

Diese Funktion übernimmt die miteinander abgestimmte gleichzeitige Verwendung zweier Sonden:

1. Ein Werkzeugschalter-Sensor, irgendwo auf der Maschine befestigt (manchmal engl. als tool-setter bezeichnet), und
2. Eine Spindelsonde, die zu Beginn des Auftrags (engl. job) vorübergehend installiert ist (manchmal als xyz-Sonde oder Renishaw-Sonde bezeichnet).

Diese Techniken sind nützlich für Maschinen, die nicht wiederholbare Werkzeughalter haben und keine automatischen Werkzeugwechselvorrichtungen aufweisen. (für Maschinen mit wiederholbaren Werkzeughaltern siehe Abschnitt auf Messwerkzeuglänge. Für Maschinen mit automatischen Werkzeugwechselgeräten, konsultieren Sie die Arbeiten im LinuxCNC-Repository bei `configs/sim/axis/remap/rack-toolchange`.)

Um diese Funktion zu nutzen, müssen Sie einige zusätzliche Einstellungen vornehmen und vielleicht möchten Sie den angebotenen HAL-Pin verwenden, um Werte in Ihrem eigenen NGC-Remap-Verfahren zu erhalten. Diese Einstellungen werden in einem späteren Abschnitt beschrieben.

Erstens, dieses Dokument deckt die Verwendung dieser Technik ab. Zweitens deckt dieses Dokument ab, wie diese Technik zu Beginn eines Produktionsablaufs vorbereitet werden kann.

Workflow Übersicht

Ein detaillierter Workflow-Durchgang folgt dieser Übersicht.

Einrichtungsschritte beinhalten:

- Eingeben der Sondengeschwindigkeiten auf der Seite mit den Sondeneinstellungen.
- Aktivieren von "Werkzeugmessung verwenden" (engl. Use Tool Measurement) auf der Versa-Probe-Registerkarte.
- Aktivieren von "Nutzen des Werkzeug Sensors" (engl. use tool sensor) bei Einstellungen (engl. settings).

IMPORTANT	Bei der Ersteinrichtung der automatischen Werkzeugmessung sollten Sie vorsichtig sein, bis Sie den Werkzeugwechsel und die Position des Messtasters bestätigt haben - ein Werkzeug/Taster kann leicht beschädigt werden. Ein Abbruch ist möglich, solange der Messtaster in Bewegung ist.

Die Werkzeugmessung in QtDragon erfolgt in den folgenden Schritten, die in dem folgenden Abschnitt genauer erklärt werden:

1. Null das Sondenwerkzeug durch Messen des tool setters mit der installierten Spindelsonde
2. Berührung des Werkstücks in X und Y.
3. Messen Sie die Höhe Ihres Blocks von der Basis, an der sich Ihr Werkzeugschalter befindet, bis zur Oberseite des Blocks (einschließlich Spannfutter usw.).
4. Geben Sie auf der Registerkarte Versa-Sonde den Messwert für die Blockhöhe ein.
5. Gehen Sie in den Automatikmodus und starten Sie Ihr Programm.

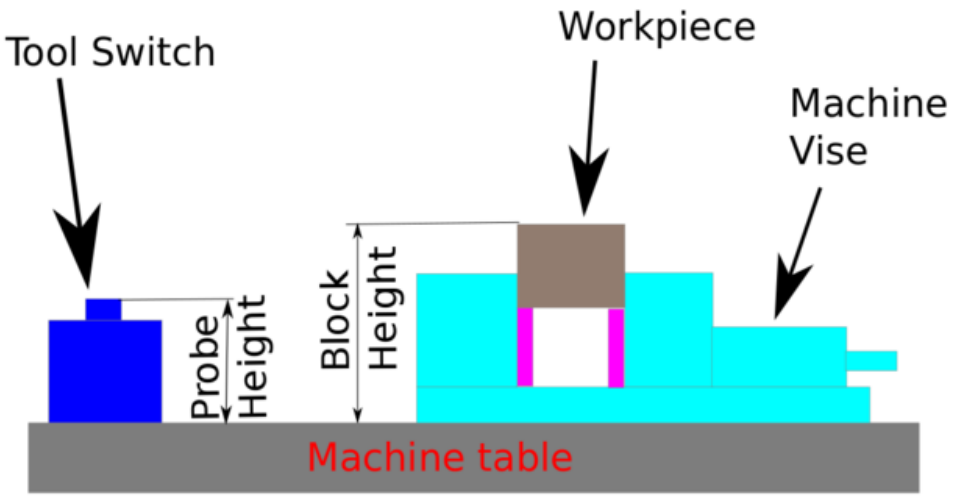


Figure 181. Automatische Werkzeugvermessung

Beim ersten gegebenen Werkzeugwechsel wird das Werkzeug vermessen und der Versatz wird automatisch auf die Blockhöhe eingestellt. Der Vorteil dieser Methode ist, dass Sie kein Referenzwerkzeug benötigen.

NOTE	Ihr Programm muss am Anfang einen Werkzeugwechsel enthalten. Das Werkzeug wird gemessen, auch wenn es schon vorher verwendet wurde, so dass keine Gefahr besteht, wenn sich die Blockhöhe geändert hat. Es gibt mehrere Videos auf you tube, die diese Technik mit GMOCCAPY demonstrieren. Der GMOCCAPY-Bildschirm war der Wegbereiter dieser Technik.

Die Reihenfolge der Ereignisse (unter Verwendung der Standarddateien in den Vorstellungen (engl. default settings)):

1. Eilgang (engl. rapid move) in Z in Position definiert im INI [TOOL_CHANGE] Z.
2. Eilgang in X und Y auf die in INI definierte Nummer [TOOL_CHANGE] X und Y.
3. Führen Sie einen normalen M6 Werkzeugwechsel durch, d.h. Stopp der Spindel, Senden einer Aufforderung an den Benutzer, das Werkzeug zu wechseln.
4. Eilgang in X und Y zu Position definiert in der INI [VERSA_TOOLSETTER] X und Y.
5. Eilgang abwärts in Z zur Position in INI [VERSA_TOOLSETTER] Z.
6. Sondieren nach unten in Z bis maximal wie definiert in der INI durch [VERSA_TOOLSETTER] MAXPROBE.
7. Ändert den Versatz (engl. offset) des aktuellen Arbeitskoordinatensystems auf die Differenz zwischen dem vorherigen Werkzeug und dem aktuell gemessenen Werkzeug.
8. Eilgang aufwärts in Z zur Position in INI [VERSA_TOOLSETTER] Z_MAX_CLEAR.
9. Schnelle Bewegung in die X Y-Position, wenn der Werkzeugwechsel aufgerufen wurde.
10. Schneller Wechsel in die Z-Position, wenn der Werkzeugwechsel aufgerufen wurde.

NOTE

Die [TOOL_CHANGE] Z-Position sollte hoch genug sein, damit das Werkzeug beim Verschieben auf die [VERSA_TOOLSETTER] X und Y Position nicht auf die Werkzeugsonde trifft.

Der MAXPROBE Abstand muss hoch genug sein, damit das Werkzeug die Sonde berühren kann.

Detailliertes Workflow-Beispiel

Einmalige Einrichtung

Folgende Setups müssen nur einmal ausgeführt werden, solange sie in Kraft bleiben:

1. Unter den Einstellungen des Messtasters (engl. "Probe Tool Screens"): Achten Sie auf angemessene Werte für „Schnell“ und „Suche“. Dies sind die Geschwindigkeiten, mit denen das Abtasten durchgeführt wird, angegeben in Maschineneinheiten pro Minute.
2. Unter den Einstellungen des Messtasters: Stellen Sie sicher, dass „Werkzeugmessung“ (engl. tool measure) aktiviert ist (der Buton muss hervorgehoben sein)
3. Bei Einstellungen: Stellen Sie sicher, dass "Nutze Werkzeug Sensor (engl. "Use Tool Sensor") aktiviert ist (die checkbox muss markiert sein)
4. In der Werkzeugtabelle: Richten Sie ein Werkzeug für den Spindelmesstaster ein und stellen Sie sicher, dass dessen Z-Offset auf null gesetzt ist.

NOTE

Es ist möglich, eine Werkzeuglänge für die Werkzeugsonde zu verwenden, die ungleich null ist. Aber dies erfordert zusätzliche Schritte und es ist leicht, dabei Fehler zu

machen. Der folgenden wird von einer Werkzeugsonden-Länge von null ausgegangen.

Prozedur vor dem Start eines Programms

Das folgende Setup wird vor dem Start eines Programms durchgeführt, das M6 Werkzeugwechselbefehle ausführt.

1. Laden Sie die Spindelsonde physisch in die Spindel.
2. Laden Sie die Spindelsonde nun auch programmtechnisch mit dem Befehl M61 Q*x* in die Spindel ein, bei dem *x* die Zahl im Werkzeugtisch für die Spindelsonde ist (innerhalb der Werkzeug-Tabellen-Registrierkarte befindet sich ein Button, der auch verwendet werden kann)
3. Positionieren über dem Werkzeugtaster: Verwenden Sie die Schaltfläche „Gehe zum Werkzeugtaster“ (engl. Go To Toolsetter) unter den Messtaster-Einstellungen, um die Spindel über dem Werkzeugtaster zu positionieren.
4. Werkzeugtaster-Messung (engl. toolsetter measure): Verwenden Sie den Button „Werkzeugtaster Z-Höhe messen“ (engl. probe tool setter Z height) unter den Messtaster-Einstellungen (engl. probe settings). Beachten Sie, dass dabei der Wert „Messtasterhöhe“ („Probe HT“) in ABS-Koordinaten ermittelt und auf dem Bildschirm „Messtaster-Einstellungen“ angezeigt wird.
5. Schnellauf (jog) zum Werkstück.
6. Werkstückmessung: Verwenden Sie die Schaltfläche „Z-Höhe des Materials messen“ unter den Messtaster-Einstellungen. Dadurch wird der Wert „Blockhöhe“ („Block Ht“) in ABS-Koordinaten ermittelt und auf dem Bildschirm „Messtaster-Einstellungen“ angezeigt. (In der Regel entspricht dieser Wert nun auch dem Z-Nullpunkt Ihres Werkstück-Koordinatensystems.)
7. Werkstück-Koordinatensystem setzen (z. B. G54 oder ein anderes): Verwenden Sie den Messtaster und das passende Messtaster-Menü oder eine andere geeignete Methode, um das für Ihre Arbeit benötigte X-, Y- und Z-Koordinatensystem festzulegen.
8. Wenn Ihr Programm mit einem TnM6-Befehl beginnt, bevor die Spindel rotiert, können Sie den Spindelmesstaster installiert lassen. Sie können auch einen TnM6-Befehl verwenden, um den Spindelmesstaster zu wechseln – wenn das Programm denselben Befehl erneut ausgibt, wird der Werkzeugwechsel übersprungen.

CAUTION Achten Sie darauf, die Spindelsonde (engl. spindle probe) nicht in der Spindel zu lassen, wenn ein Programm die Spindel starten kann.

Sobald diese Schritte abgeschlossen sind, kann ein Programm mit mehreren T*n*M6 Werkzeugwechseln gestartet werden und wird mit automatischen Pausen für manuelle Werkzeugwechsel mit anschließender automatisierter Werkzeugmessung arbeiten.

NOTE Nachdem die neue Werkzeuglänge mit dem Werkzeugtaster ermittelt wurde, verwendet dieser Remap-Code ein G43, das den Versatz auf das Werkstück-Koordinatensystem anwendet, das beim Aufruf des M6-Befehls aktiv war. Da das Remapping das Werkstück-Koordinatensystem um den Versatz zwischen dem vorherigen und dem aktuellen Werkzeug angepasst hat, wird die Werkzeugspitze an genau der gleichen Position im Raum enden, an der sich die Spitze des vorherigen Werkzeugs beim Aufruf des

Werkzeugwechsels befand.

Ertasten der Werkstückhöhe in QtDragon_hd

Die [TOOL_CHANGE] Z-Position sollte hoch genug sein, damit das Werkzeug beim Verschieben auf die [VERSA_TOOLSETTER] X und Y Position nicht auf die Werkzeugsonde trifft. Der MAXPROBE Abstand muss hoch genug sein, um die Sonde zu berühren.

Werkstückhöhe Antasten

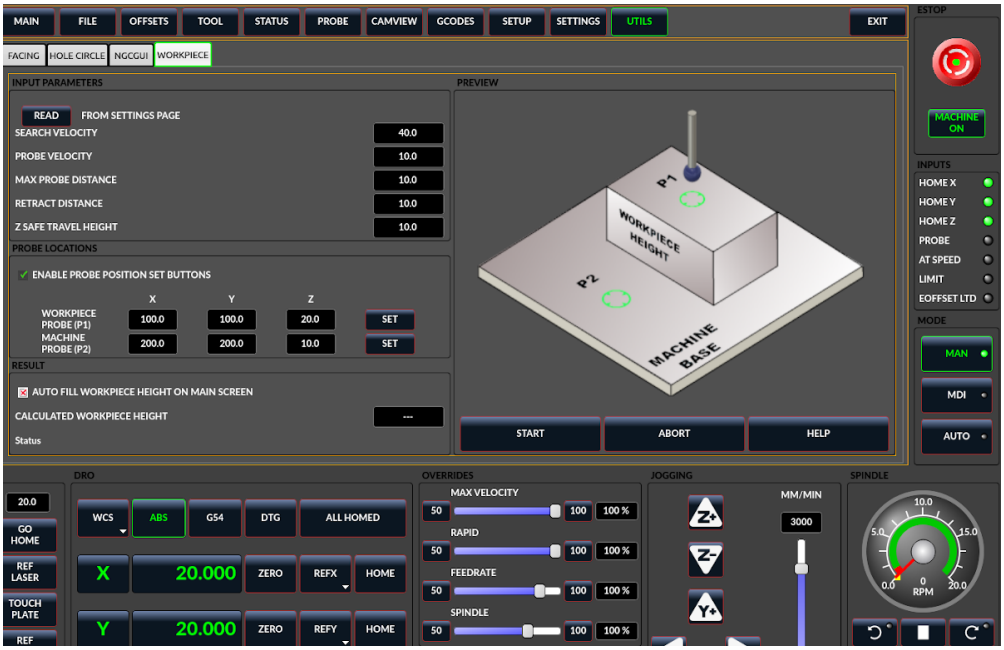


Figure 182. QtDragon_hd - Werkstück Höhenabtastung

Dieses Programm tastet 2 benutzerdefinierte Positionen in der Z-Achse an und berechnet die Höhendifferenz.

NOTE Diese Funktion ist nur in der Version QtDragon_hd verfügbar.

Buttons zum Einstellen der Sondenposition aktivieren

- Wenn diese Option markiert ist, sind die SET-Tasten aktiviert.
- Dadurch kann der Benutzer die X-, Y- und Z-Parameter automatisch mit der aktuellen Position, wie sie auf den DROs angezeigt wird, ausfüllen.

Automatische Füllung der Werkstückhöhe auf dem Hauptbildschirm

- Wenn diese Option aktiviert ist, wird die berechnete Höhe automatisch in das Feld Werkstück-Höhe im Hauptbildschirm übertragen.
- Andernfalls ist der Hauptbildschirm nicht betroffen.

Werkstücktaster bei

- die X-, Y- und Z-Koordinaten geben an, wo die erste Sondierungsroutine im aktuellen WCS beginnen soll

Probenahme bei

- die X-, Y- und Z-Koordinaten geben an, wo die zweite Sondierungsroutine beginnen soll, und zwar im aktuellen WCS

Z Sichere Fahrhöhe

- Die Maschine wird auf die sichere Z-Fahrhöhe angehoben, bevor sie zu den X- und Y-Koordinaten rüttelt.
- Die Spindel senkt sich dann auf die angegebene Z-Koordinate.
- Sie sollte so gewählt werden, dass das Werkzeug beim Joggen alle Hindernisse überwindet.

START-Button

- Die Maschine fährt zur ersten Position und tastet dann nach unten.
- Die Maschine fährt dann zum zweiten Standort und tastet sich erneut nach unten.
- Die Differenz der angetasteten Werte wird als berechnete Werkstückhöhe angegeben.
- Die Parameter für Suchgeschwindigkeit, Sondengeschwindigkeit, maximale Sondenentfernung und Rücklaufentfernung werden von der Hauptseite der GUI-Einstellungen gelesen.

ABORT-Button

- bewirkt, dass alle derzeit ausgeführten Tipp- und Tastroutinen gestoppt werden.

HELP-Button

- zeigt dieses Hilfefenster an.

NOTE

- Zwei beliebige Punkte können im Maschinenarbeitsraum spezifiziert werden.
- Wenn der erste Punkt höher ist als der zweite, ist die berechnete Höhe eine positive Zahl.
- Ist der erste Punkt niedriger als der zweite, ist die berechnete Höhe eine negative Zahl.
- Die Einheiten sind in diesem Programm irrelevant. Die gemessenen Werte werden nicht gespeichert und nur die Differenz wird gemeldet.

CAUTION

Das Einstellen falscher Werte kann zu Abstürzen in Vorrichtungen auf der Arbeitsfläche der Maschine führen. Ein erster Test ohne Werkzeug und in sicherer Höhe wird empfohlen.

Werkzeugmess-Pins

Versaprobe bietet 5 Ausgabepins für die Werkzeugmessung. Die Pins werden von einem Remap-G-Code-Unterprogramm gelesen, so dass der Code auf verschiedene Werte reagieren kann.

`qtversaprobe.enable` (HAL_BOOL)

Messung aktiviert oder nicht Werkzeug. Spiegelt den Zustand des Bildschirmtaste (engl. screen button) wider.

qtversaprobe.blockheight (HAL_REAL)

Die gemessene Höhe der Oberseite des Werkstücks. Spiegelt die Eingabe am Bildschirm wider.

qtversaprobe.probeheight (HAL_REAL)

Die Höhe des Schalters für die Werkzeugsonde. Spiegelt die Eingabe am Bildschirm wider.

qtversaprobe.searchvel (HAL_REAL)

Die Geschwindigkeit, mit der nach dem Schalter für den Werkzeugmesstaster gesucht wird

qtversaprobe.probevel (HAL_REAL)

Die Länge des Werkzeugs zwischen Geschwindigkeit und Messtaster. Spiegelt die Eingabe am Bildschirm wider.

qtversaprobe.backoffdist (HAL_REAL)

Die Entfernung, um die sich die Sonde nach dem Auslösen zurückzieht. Entsprich der Bildschirmeingabe.

Änderungen an der INI-Datei für Werkzeugmessungen

Ändern Sie Ihre INI-Datei so, dass sie Folgendes enthält:

Der Abschnitt PROBE (Werkstück-Sensor)

QtDragon ermöglicht es Ihnen, eine von zwei Arten von Messtasterroutinen auszuwählen. Versa Probe arbeitet mit einem M6-Remap, um die automatische Werkzeugerfassung hinzuzufügen.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

Der RS274NGC-Abschnitt

RS274NGC Section Details
Remap Statement Details
Remap Abort Details

NOTE Diese Standardeinträge sollten in den meisten Situationen gut funktionieren. Einige Systeme müssen möglicherweise anstelle von *linuxcnc/nc_files/examples/* dann *linuxcnc/nc_files/* verwenden. Bitte überprüfen Sie, ob Pfade gültig sind. Benutzerdefinierte Einträge mit Referenzen auf die geänderte Datei sind möglich.

```
[RS274NGC]

# diese Pfade so anpassen, dass sie auf Ordner mit stdglu.py und qt_auto_tool_probe.ngc
zeigen
# oder ähnlich kodierte benutzerdefinierte Remap-Dateien
SUBROUTINE_PATH = ~/linuxcnc/nc_files/remap-subroutines:\
~/linuxcnc/nc_files/remap_lib
```

```
# ist die Sub, die aufgerufen wird, wenn ein Fehler beim Werkzeugwechsel auftritt.
ON_ABORT_COMMAND=O <on_abort> Aufruf

# Der Remap-Code für die automatische Werkzeugsonde von Z der Versaprobe von QtVCP
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=qt_auto_probe_tool epilog=change_epilog
```

Der Abschnitt Werkzeugsensor (engl. tool sensor)

Die Position des Werkzeugsensors und die Startposition der Antastbewegung.
Alle Werte sind absolute (G53) Koordinaten, mit Ausnahme von MAXPROBE, das als absolute Länge der Bewegung angegeben wird.
Alle Werte sind in maschineneigenen Einheiten.
X,Y,Z setzen die Lokalisation der Sonde für den Werkzeug-Einsteller (engl. tool setter probe location).

Ablauf der automatischen Tastaktion im standardmäßigen Beispiel-Remap qt_auto_probe_tool (dieses Verhalten kann durch Ändern entweder der remap-Anweisung im RS274NGC-Abschnitt oder durch Modifikation des Codes in qt_auto_probe_tool.ngc geändert werden):

1. Schnellbewegung zur Z-Position aus der [CHANGE_POSITION]-Sektion der INI-Datei (dies ist eine relative Bewegung; der Z-Wert wird zur aktuellen Z-Koordinate addiert)
2. schnelle (engl. rapid) Bewegung zur X & Y Position der [CHANGE_POSITION] aus der INI Datei.
3. warten auf manuelle Bestätigung des werkzeugwechsels
4. schnelle Bewegung zur [VERSA_TOOLSETTER] X & Y Position
5. schnelle Bewegung zur [VERSA_TOOLSETTER] Z_MAX_CLEAR Z-Position
6. schnelle Sondierung
7. langsame Sondierung
8. schnelle Bewegung zur [VERSA_TOOLSETTER] Z_MAX_CLEAR Z-Position

Z_MAX_CLEAR ist die Z-Position, zu der sich das System begibt bevor der Tool-Setter bewegt wird, wenn Sie die *Travel to Toolsetter-Taste* betätigen.
Travel to Toolsetter Aktionssequenz:

1. schnelle Bewegung zur [VERSA_TOOLSETTER] Z_MAX_CLEAR Z-Position
2. schnelle Bewergung zur [VERSA_TOOLSETTER] XY Position
3. schnelle Bewegung zur [VERSA_TOOLSETTER] Z Position.

Beispiel-Einstellungen:

```
[VERSA_TOOLSETTER]
X = 10
Y = 10
Z = -20
Z_MAX_CLEAR = -2
MAXPROBE =  -20
```

Der Abschnitt "Position ändern"

Die Position wird nicht zweckdienlichst TOOL_CHANGE_POSITION genannt - **canon verwendet diesen Namen und interveniert sonst.** Die Position, an welche die Maschine bewegt werden soll, bevor der Befehl zum Werkzeugwechsel gegeben wird. Alle Werte sind in absoluten Koordinaten. Alle Werte sind in maschineneigenen Einheiten.

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

Die Python-Sektion

Der Python-Abschnitt legt fest, nach welchen Dateien der Python-Interpreter von LinuxCNC sucht, z.B. die Datei `toplevel.py` im Ordner `python` im Konfigurationsverzeichnis: Diese Standardeinträge sollten in den meisten Situationen gut funktionieren. Einige Systeme müssen möglicherweise *linuxcnc/nc_files/examples/* verwenden statt *linuxcnc/nc_files*. Benutzerdefinierte Einträge, die auf die geänderte Datei verweisen, sind möglich.

```
# The path start point for all remap searches, i.e. Python's sys.path.append()
PATH_APPEND = ~/linuxcnc/nc_files/remap_lib/python-stdglue/python
# path to the tremap's 'toplevel file
TOPLEVEL = ~/linuxcnc/nc_files/remap_lib/python-stdglue/python/toplevel.py
```

Benötigte HAL-Verbindungen

Stellen Sie sicher, dass der Messtastereingang in Ihrer HAL-Datei angeschlossen ist: Bei korrektem Anschluss sollten Sie in der Lage sein, die Taster-LED in QtDragon umzuschalten, wenn Sie den Tasterstift drücken.

```
net probe motion.probe-input <= <Ihr_input_pin>
```

10.5.15. Ausführen von gegebener Zeile

Ein G-Code-Programm kann an jeder beliebigen Zeile gestartet werden, indem Sie im AUTO-Modus auf die gewünschte Zeile in der G-Code-Anzeige klicken. Es liegt in der Verantwortung des Bedieners sicherzustellen, dass sich die Maschine im gewünschten Betriebsmodus befindet. Es wird ein Dialogfeld angezeigt, in dem die Spindelrichtung und -geschwindigkeit voreingestellt werden können. Die Startlinie wird in dem Feld mit der Bezeichnung LINE neben der Taste CYCLE START angezeigt. Die Funktion "Von der Linie starten" kann auf der Einstellungsseite deaktiviert werden.

NOTE	LinuxCNC's "Ausführen ab Zeile..." (engl. run-from-line) ist nicht sehr benutzerfreundlich. Z.B. startet es nicht die Spindel oder bestätigt das richtige Werkzeug. Auch werden Unterprogramme nicht gut gehandhabt. Wenn es verwendet wird, ist es am besten, mit einem Eilgang zu beginnen.

10.5.16. Laser-Buttons

Der Button LASER ON/OFF dient dazu, einen Ausgang ein- oder auszuschalten, der mit einem kleinen Laserkreuzprojektor verbunden ist. Wenn das Fadenkreuz über einem gewünschten Referenzpunkt auf dem Werkstück positioniert ist, kann die Taste REF LASER gedrückt werden, die dann die X- und Y-Offsets auf die in den Feldern LASER OFFSET auf der Seite Einstellungen angegebenen Werte setzt.

10.5.17. Beschreibung der Registerkarten

Tabs allow the user to select the most appropriate info/control on the top three panels. If the on screen keyboard is showing and the user wishes to hide it but keep the current tab, they can do that by pressing the *HIDE* button on the virtual keyboard. In QtDragon, there is a splitter handle between the G-code text display and the G-code graphical display. One can use this to split the size between the two areas. This can be set differently in each tab and in each mode. The positions will be remembered.

Hauptregisterkarte

Auf dieser Registerkarte wird die grafische Darstellung des aktuellen Programms angezeigt. Mit den seitlichen Schaltflächen wird die Anzeige gesteuert.

- *Benutzeransicht*: Auswählen/Wiederherstellen einer benutzerdefinierten Ansicht des aktuellen Programms.
- *P*,*X*,*Y*,*Z*: Standardansichten einstellen.
- *D*: Anzeige der Abmessungen umschalten.
- +, -: Zoom-Steuerung.
- *C*: Übersichtliche Grafik der Werkzeugbewegungslinien.

In `qtdragon_hd` sind auf der rechten Seite auch Makrotasten verfügbar. Bis zu zehn Buttons können in der INI definiert werden.

Registerkarte "Datei"

Sie können diese Registerkarte verwenden, um Programme zu laden oder zu übertragen. Die Bearbeitung von G-Code-Programmen kann über diese Registerkarte ausgewählt werden. Bei `qtdragon_hd` können Sie hier den *G-Code Ripper* laden.

Registerkarte "Offsets"

Auf dieser Registerkarte können Sie die System-Offsets überwachen/ändern. Es gibt praktische Schaltflächen für die Nullstellung der Rotation.G92 und den aktuellen G5x-Benutzer-Offset.

Registerkarte "Werkzeug"

Auf dieser Registerkarte können Sie die Werkzeugkorrekturen überwachen/verändern. Das Hinzufügen und Löschen von Werkzeugen aus der Werkzeugdatei kann ebenfalls über diese Registerkarte erfolgen. Wenn diese Registerkarte ausgewählt ist, ändern sich die einzelnen Home-Schaltflächen im DRO-Bereich

in Schaltflächen zur Einstellung der Werkzeugkorrektur. Sie kehren zu den Home-Schaltflächen zurück, wenn Sie eine andere Registerkarte auswählen. Wenn Sie diese Schaltfläche drücken, wird ein Menü mit Optionen angezeigt:

- Aktuelle Werkzeug-Position einstellen
- Aktuelle Werkzeug-Position anpassen
- Aktuelle Werkzeug-Position auf Null setzen
- Werkzeugversatz direkt einstellen
- Auf zuletzt zurücksetzen

Registerkarte "Status"

Hier wird ein mit einem Zeitstempel versehenes Protokoll wichtiger Maschinen- oder Systemereignisse angezeigt. Maschinenereignisse sind eher für einen Bediener geeignet, während die Systemereignisse bei der Fehlersuche helfen können.

Registerkarte "Sonde

Auf dieser Registerkarte werden Optionen für Prüfroutinen angezeigt. Abhängig von den INI-Optionen kann dies VersaProbe- oder BasicProbe-Stil sein. Sie sind funktionell ähnlich. QtDragon_hd zeigt auch ein kleineres Grafikanzeigefenster an.

Camview-Registerkarte

Wenn die erkannte Webcam angeschlossen ist, wird auf dieser Registerkarte das Videobild mit einem Fadenkreuz überlagert angezeigt einem Kreis und einer Gradanzeige überlagert. Dies kann angepasst werden, um ein Teilmerkmal für solche Dinge wie Touchoff anzupassen. Die zugrunde liegende Bibliothek verwendet das openCV-Python-Modul, um eine Verbindung zur Webcam herzustellen.
Um das Seitenverhältnis der X- oder Y-Größe (in Prozent), die Kamera-Portnummer, das API-Backend oder die angeforderte Auflösung anzupassen, suchen Sie in der Einstellungen (engl. preferences)-Datei nach:

```
[CUSTOM_FORM_ENTRIES]
Camview xscale = 100
Camview yscale = 100
Camview cam number = 0
Camview cam api = V4L2
Camview cam resolution = 1280,720
```

Diese Angaben sind in Prozent, normalerweise wird der Bereich in einer Achse 100 - 200 betragen.
Eine Invertierung dieser Skalierung kann genutzt werden, um das Bild in X, Y oder beiden Achsen zu spiegeln.
Die API stammt aus OpenCV; die verfügbaren Backends werden aufgelistet, wenn die Option -V für Debugging verwendet wird. Mit der Einstellung „ANY“ überlässt man OpenCV die Auswahl.
Setzen Sie die Auflösung auf „DEFAULT“, damit OpenCV die passende wählt. Verfügbaren Auflösungen werden angezeigt, wenn -V für Debugging aktiviert ist.

NOTE Die Einstellungen (engl. preferences) Datei kann nur editiert werden während QtDragon nicht läuft.

G-Codes Registrierkarte

Diese Registerkarte zeigt eine Liste von LinuxCNC's G-Code. Wenn Sie auf eine Zeile klicken, wird eine Beschreibung des Codes angezeigt.

Registerkarte "Einstellungen"

Es ist möglich, eine HTML- oder PDF-Datei (.html / .pdf-Endung) mit Einrichtungshinweisen zu laden, und diese werden auf der Registerkarte "Setup" angezeigt.
Wenn Sie ein G-Code-Programm laden und es eine HTML/PDF-Datei mit demselben Namen gibt, wird es automatisch geladen.
Einige Programme wie Fusion 360 und Aspire erstellen diese Dateien für Sie. Sie können auch Ihre eigenen HTML-Docs mit der enthaltenen SetUp Writer-Taste schreiben.
Es gibt drei Unterkarten (engl. tabs):

- *HTML* - alle geladenen HTML-Seiten werden hier angezeigt. Die Navigations-Buttons funktionieren auf dieser Seite.
- *PDF* - alle geladenen PDF Setup-Seiten werden hier angezeigt.
- *PROPERTIES* - wenn ein Programm geladen wird, werden hier dessen G-Code Eigenschaften angezeigt.

Es gibt Navigations-Buttons für die HTML-Seite:

- Der Pfeil nach oben führt Sie auf die Standard-HTML-Seite zurück.
- Der linke Pfeil führt auf die vorherige HTML-Seite.
- Der rechte Pfeil revidiert die Wirkung eines zuvor gedrückten Pfeils nach links.

Wenn Sie eine benutzerdefinierte Standard-HTML-Seite einbinden möchten, nennen Sie diese *default_setup.html* und legen Sie es in Ihren Konfigurationsordner.
In dieser Registerkarte können benutzerdefinierte QtVCP-Panels angezeigt werden, indem die Option EMBED_TAB_LOCATION auf *tabWidget_setup* gesetzt wird.



Figure 183. QtDragon - Beispiel für die Registerkarte Setup

Registerkarte "Einstellungen"

Die Registerkarte "Einstellungen" dient zum Einstellen der Betriebsoptionen, der Offsets für Messtaster/Tastplatte/Laser/Kamera und zum Laden externer Debugging-Programme.

Registerkarte "Dienstprogramme"

Auf dieser Registerkarte wird eine weitere Auswahl von G-Code-Hilfsprogrammen (und allen eingebetteten Panels) angezeigt:

- *Facing*: ermöglicht schnelles Planfräsen eines definierbaren Bereichs in Winkeln von 0,45 und 90 Grad.
- *Lochkreis*: ermöglicht die schnelle Einstellung eines Programms zum Bohren eines Lochkreises mit definierbarem Durchmesser und Anzahl der Löcher.
- *NGCGUI*: ist eine QtVCP-Version des beliebten G-Code Subroutine Builder/Selector, siehe Widgets-NGCGUI.
- *Hole Enlarge*: allows milling a preexisting hole larger.

These tabs are detachable from the main screen by pressing the small arrow key on the far right of the tab header.
You can close the panel by pressing the arrow again or closing the window with the x button.
The last size and location of the detached window will be remembered each time the window is closed.
Custom QtVCP panels can be displayed here by setting the EMBED_TAB_LOCATION option to `tabWidget_utilities`

Benutzer-Registerkarte

Diese Registerkarte wird nur angezeigt, wenn ein eingebettetes Panel für die Position `stackedWidget_mainTab` bestimmt wurde. Wenn mehr als eine eingebettete Registerkarte festgelegt wurde, werden diese durch Drücken der Benutzer-Registerkarte durchlaufen.

10.5.18. Shutdown Option

If you desire, it is possible to have the shutdown dialog close the screen after a timed countdown.
You can edit the preference file to set the length of the countdown in seconds.
One can only edit the preference file with the screen unloaded.
A setting of 0 (default) will wait indefinitely.

Look in the preference file for:

```
[SHUTDOWN_OPTIONS]
auto_shutdown_timeout = 6
```

10.5.19. Stile

Nahezu alle Aspekte des Erscheinungsbildes der Benutzeroberfläche sind über die Stylesheet-Datei QtDragon.qss konfigurierbar. Die Datei kann manuell oder über das Stylesheet-Dialog-Widget in der GUI bearbeitet werden. Um diesen Dialog aufzurufen, drücken Sie F12 im Hauptfenster. Neue Stile können vorübergehend angewendet und dann in einer neuen qss-Datei gespeichert werden, oder die aktuelle qss-Datei überschreiben.

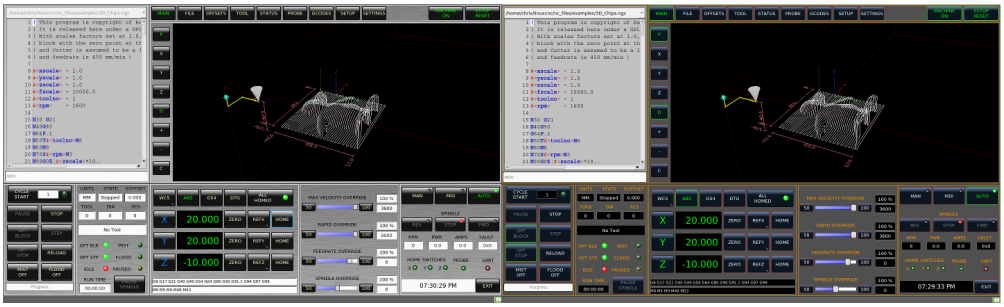


Figure 184. QtDragon - Zwei Stil-Beispiele

10.5.20. Internationalisierung

Es ist möglich, Übersetzungsdateien für QtDragon zu erstellen, um sie in der Sprache des aktuellen Gebietsschemas (engl. locale) anzuzeigen.

Um eine Übersetzungsdatei zu erstellen und oder zu bearbeiten, muss LinuxCNC installiert sein und vor Ort ausgeführt werden.

Im Folgenden wird davon ausgegangen, dass das LinuxCNC git-Verzeichnis ~/linuxcnc-dev ist.

NOTE	Wenn Sie QtDragon_hd nutzen, ersetzen Sie *qtdragon* mit *qtdragon_hd*

Alle Sprachdateien werden in ~/linuxcnc-dev/share/screens/qtdragon/languages gespeichert.

Die Datei qtdragon.py ist eine Python-Version der GUI-Datei, die für Übersetzungszwecke verwendet wird.

Die .ts-Dateien sind die Quelldateien für die Übersetzungen. Dies sind die Dateien, die für jede Sprache erstellt/bearbeitet werden müssen.

Die .qm-Dateien sind die kompilierten Übersetzungsdateien, die von pyqt verwendet werden.

Die Sprache wird durch einen Unterstrich plus die ersten beiden Buchstaben des Gebietsschemas bestimmt, z. B. bei einer italienischen Übersetzung wäre dies `_it`. In diesem Dokument wird sie mit `_xx` bezeichnet, so dass `qtdragon_xx.ts` in diesem Dokument für eine italienische Übersetzung eigentlich `qtdragon_it.ts` wäre.

Das voreingestellte Gebietsschema für QtDragon ist `_en`, was bedeutet, dass Übersetzungsdateien, die als `qtdragon_en.*` erstellt wurden, nicht für Übersetzungen verwendet werden.

Wenn eines der erforderlichen Dienstprogramme (pyuic5, pylupdate5, linguist) nicht installiert ist, muss der Benutzer die benötigten Software-Tools zur Entwicklung installieren:

```
sudo apt install qttools5-dev-tools pyqt5-dev-tools
```

Wechseln Sie in das Sprachenverzeichnis:

```
cd ~/linuxcnc-dev/share/qtvcp/screens/qtdragon/languages
```

Wenn Textänderungen an der grafischen Benutzeroberfläche vorgenommen wurden, führen Sie den folgenden Befehl aus, um die GUI-Python-Datei zu aktualisieren:

```
pyuic5 ../qtdragon.ui > qtdragon.py
```

Der Benutzer kann entweder eine neue Übersetzungsquelldatei für eine nicht existierende Sprachübersetzung erstellen oder eine existierende Übersetzungsquelldatei modifizieren, wenn ein Text in einer QtDragon-Quelldatei geändert wurde. Wenn Sie eine bestehende Übersetzung modifizieren, die keine Änderungen in der Quelldatei erfahren hat, ist dieser Schritt nicht erforderlich.

Erstellen oder bearbeiten Sie eine .ts-Datei:

```
./langfile xx
```

NOTE Dieser Befehl ist ein Skript, das Folgendes ausführt: $ pylupdate5 **.py** **..**/.py ../../../../../lib/python/qtvcp/lib/qtdragon/*.py -ts qtdragon_xx.ts

Die Bearbeitung der Übersetzung erfolgt mit der Anwendung Linguist:

```
linguist
```

1. Öffnen Sie die TS-Datei und übersetzen Sie die Zeichenfolgen

Es ist nicht notwendig, für jede Textzeichenfolge eine Übersetzung bereitzustellen. Wenn für eine Zeichenfolge keine Übersetzung angegeben ist, wird die ursprüngliche Zeichenfolge in der Anwendung verwendet. Der Benutzer muss auf die Länge der Zeichenketten achten, die in den Widgets erscheinen, da der Platz begrenzt ist. Wenn möglich, sollte die Übersetzung nicht länger als das Original sein.

Wenn die Bearbeitung abgeschlossen ist, speichern Sie die Datei:
`Datei -> Speichern` (engl. File → Save)

Erstellen Sie dann die .qm-Datei:
`Datei -> Freigabe` (engl. `File -> Release`)

QtDragon wird beim nächsten Start in die Sprache des aktuellen Gebietsschemas übersetzt, solange eine .qm Datei in dieser Sprache existiert.

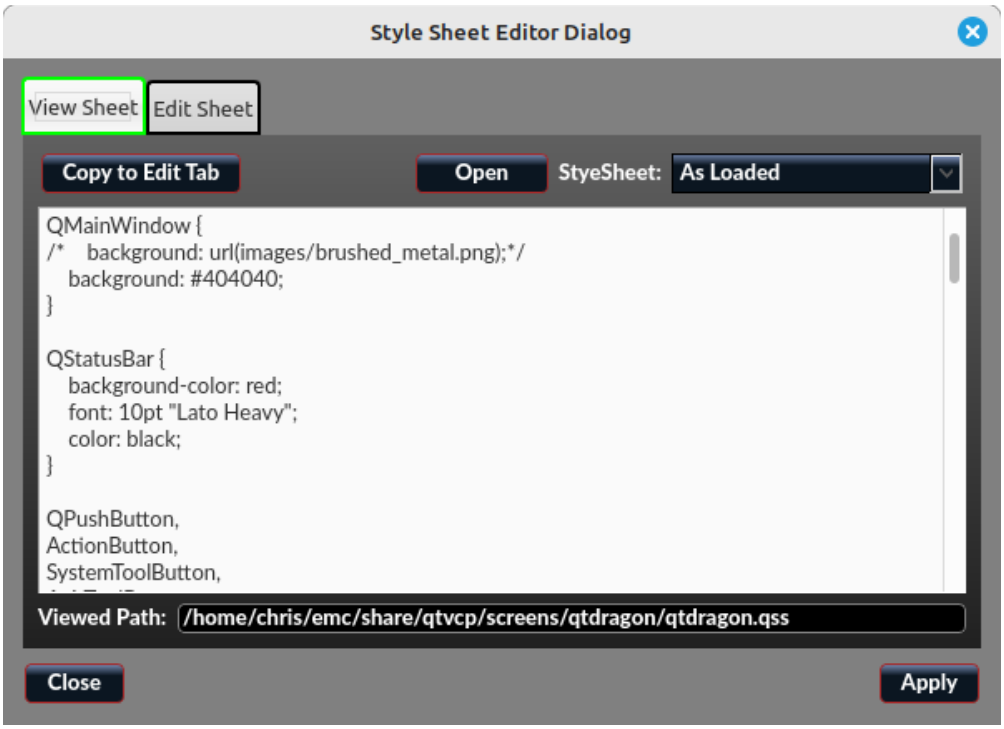
Benutzer sind herzlich eingeladen, Übersetzungsdateien zur Aufnahme in QtDragon einzureichen. Die hierfür bevorzugte Methode ist das Erstellen eines Pull Requests vom eigenen GitHub-Konto, wie an anderer Stelle beschrieben. Die einzigen Dateien, die eingereicht werden müssen, sind qtdragon_xx.ts und qtdragon_xx.qm.

10.5.21. Anpassung

Ein allgemeiner Überblick über die Anpassung mitgelieferter Bildschirmmasken.

Stylesheets

Stylesheets können für eine Vielzahl von Anpassungen genutzt werden, erfordern jedoch in der Regel Grundkenntnisse über die Namen der Widgets. Durch Drücken von **F12** wird ein Stylesheet-Editor-Dialog geöffnet, in dem Änderungen geladen, getestet und gespeichert werden können.
Der Tab **View Sheet** ermöglicht die Auswahl und Anwendung eines Stylesheets, das QtDragon beim ersten Laden verwenden soll.
Mit der Schaltfläche **Copy to Edit Tab** kann das aktuelle Stylesheet in den Bearbeitungs-Tab kopiert werden.
Der Tab **Edit Sheet** erlaubt das Bearbeiten, Anwenden und Speichern von Änderungen am angezeigten Stylesheet.



Manchmal sind diese Zeilen bereits vorhanden und können geändert werden, andernfalls müssen sie hinzugefügt werden.

Um zum Beispiel die DRO-Schriftart zu ändern (suchen diesem Eintrag und ändern des Namens der Schriftart):

```
DROLabel,
StatusLabel#status_rpm {
    border: 1px solid black;
    border-radius: 4px;
    font: 20pt "Noto Mono";
}
```

Um das DRO-Display-Schrift- und Anzeigeformat zu ändern:

```
DROLabel {
    font: 25pt "Lato Heavy";
    qproperty-imperial_template: '%9.5f';
    qproperty-metric_template: '%10.4f';
    qproperty-angular_template: '%11.2f';

    /*Anpassen der the Menü Optionen */
    qproperty-showLast: true;
    qproperty-showDivide : false;
    qproperty-showGotoOrigin: false;
    qproperty-showZeroOrigin: false;
    qproperty-showSetOrigin: true;
    qproperty-dialogName: CALCULATOR;
}
```

Ändern der Menüeinträge des Klickmenüs der Achsauswahl-Schaltfläche:

```
AxisToolButton {
    /* Passe all die Menü-Optionen an */
    qproperty-showLast: false;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: false;
    qproperty-dialog_code_string: CALCULATOR;
}
```

So ändern Sie den Text der Schaltfläche "Nebel" in "Luft" (fügen Sie diese Zeilen ein)

```
#action_mist{
    qproperty-true_state_string: "Air\\nOn";
    qproperty-false_state_string: "Air\\nOff";
}
```

Um die Offsets-Anzeige Schriftart und Format zu ändern:

```
ToolOffsetView {
```

```
    font: 20pt "Lato Heavy";
    qproperty-imperial_template: '%9.1f';
    qproperty-metric_template: '%10.1f';

}

OriginOffsetView {
    font: 12pt "Lato Heavy";
    qproperty-imperial_template: '%9.1f';
    qproperty-metric_template: '%10.1f';
}
```

Um den Unschärfeeffekt bei Dialogen zu beenden:

```
#screen_options {
    qproperty-focusBlur_option: false;
}
```

Um Status-Highlight/Auswahl Farben zu ändern:

```
#screen_options {
    qproperty-user1Color: white;     /*  regulärer status     */
    qproperty-user2Color: #ff9000;   /*  warnung liegt vor     */
    qproperty-user3Color: #ff8a96;   /*  kritischer Status    */
    qproperty-user4Color: #ffaa00;   /*  Handgerät Auswahl         */
    qproperty-user5Color: #ff0000;   /*  Zyklus Start Auswahl */
}
```

Ändern der G-Code Text Display Farben/Schriften:

```
EditorBase{
background:black;
qproperty-styleColorCursor:white;
qproperty-styleColorBackground:grey;
qproperty-styleColor0: black;
qproperty-styleColor1: darkblue;
qproperty-styleColor2: blue;
qproperty-styleColor3: red;
qproperty-styleColor4: lightblue;
qproperty-styleColor5: white;
qproperty-styleColor6: lightGreen;
qproperty-styleColor7: yellow ;
qproperty-styleColorSelectionText: white;
qproperty-styleColorSelectionBackground: blue;
qproperty-styleFont0: "Times,15,-1,5,90,0,0,1,1,0";
qproperty-styleFont1: "Times,15,-1,5,90,1,0,1,0,0";
qproperty-styleFont2: "Times,15,-1,5,90,0,0,1,1,0";
qproperty-styleFont3: "Times,15,-1,5,90,0,0,1,1,0";
qproperty-styleFont4: "Times,15,-1,5,90,0,0,1,1,0";
qproperty-styleFont5: "Times,15,-1,5,90,0,0,1,1,0";
qproperty-styleFont6: "Times,15,-1,5,90,0,0,1,1,0";
qproperty-styleFont7: "Times,15,-1,5,90,0,0,1,1,0";
}
```

Damit die manuellen Spindeltasten auch die Geschwindigkeit schrittweise erhöhen/verringern:

```
#action_spindle_fwd{
    qproperty-spindle_up_action: true;
}
#action_spindle_rev{
    qproperty-spindle_down_action: true;
}
```

Qt Designer und Python-Code

Alle Aspekte der grafischen Benutzeroberfläche können über Qt Designer und/oder Python-Code vollständig angepasst werden.
Diese Fähigkeit ist in der QtVCP-Entwicklungsumgebung enthalten.
Die umfangreiche Verwendung von QtVCP-Widgets hält die Menge des erforderlichen Python-Codes auf ein Minimum und ermöglicht relativ einfache Änderungen.
Auf der LinuxCNC-Website finden Sie eine ausführliche Dokumentation über die Installation und Verwendung der QtVCP-Bibliotheken.
Siehe QtVCP für weitere Informationen zu QtVCP im allgemeinen.
Individuelle Anpassungen können hinzugefügt werden durch Spezialisierungen (engl. *subclassing*) in der handler Datei. Dies fügt Quellcode dem Original hinzu.
QtDragon kann auch QtVCPs rc-Datei verwenden, um kleinere python-Code-Änderungen vorzunehmen, ohne eine benutzerdefinierte Handler-Datei.

Siehe Bildschirmmasken anpassen für weitere Informationen zur Anpassung.

Einige Widget Anpassungen sind verfügbar für basic probe und versa probe.

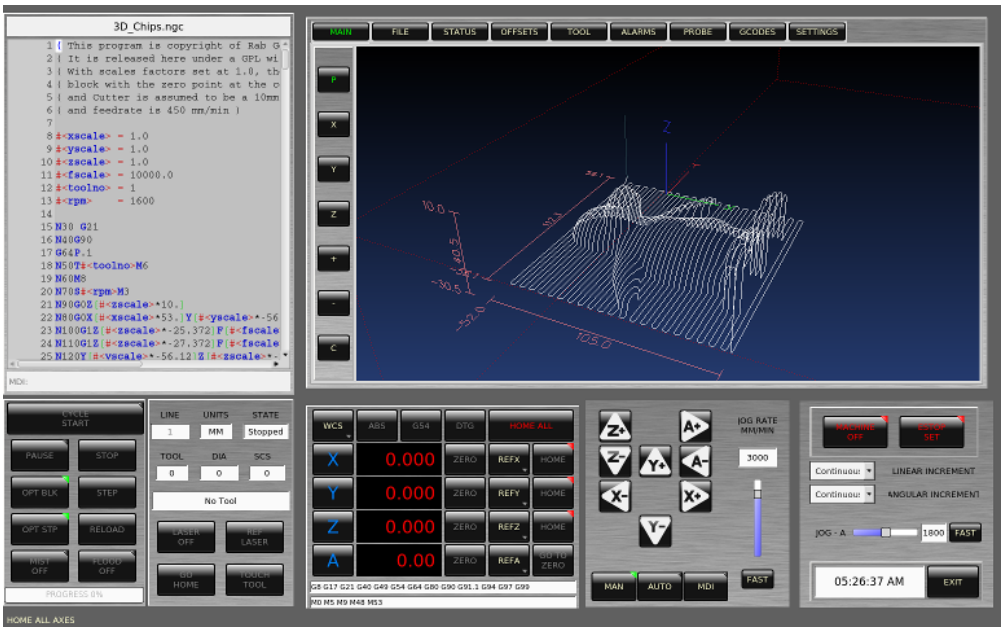


Figure 185. QtDragon - Angepasster QtDragon

10.6. NGCGUI

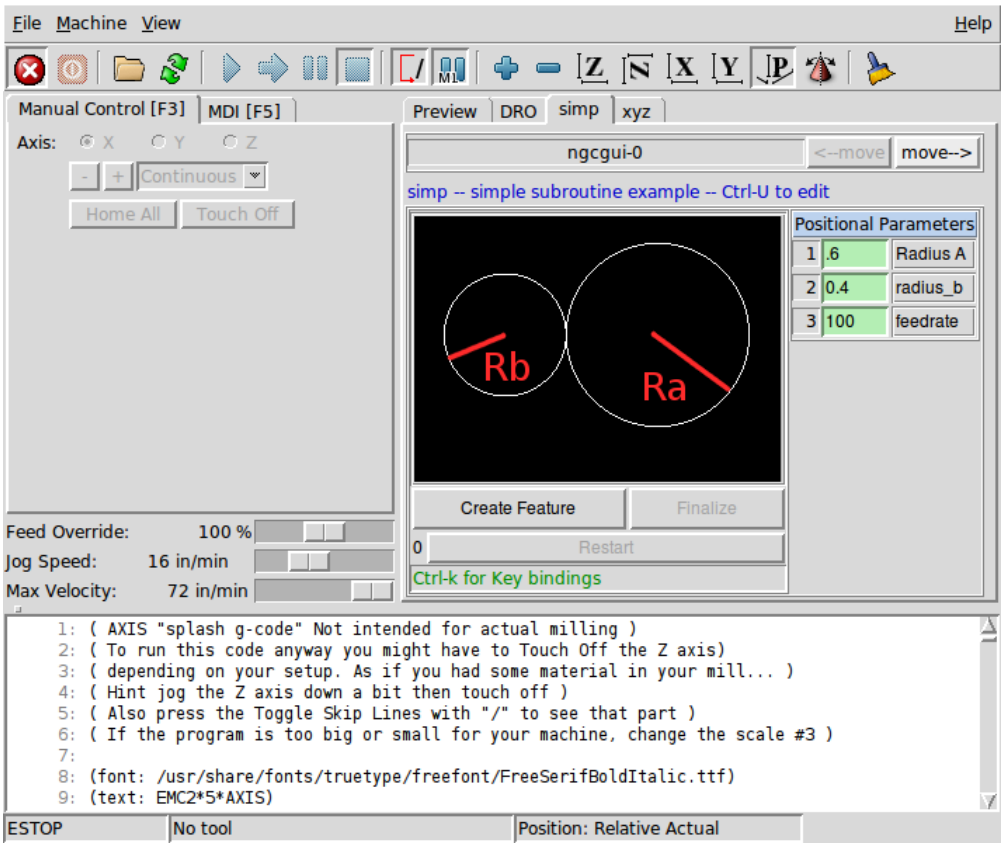


Figure 186. NGCGUI eingebettet in AXIS

10.6.1. Übersicht

- *NGCGUI* ist eine Tcl-Anwendung zur Arbeit mit Unterroutinen. Es ermöglicht Ihnen, eine Konversationsschnittstelle mit LinuxCNC zu haben. Sie können die Unterroutinen in der Reihenfolge organisieren, in der Sie sie ausführen und die Unterroutinen in einer Datei für ein vollständiges Teileprogramm verketten müssen.
- *NGCGUI* kann als eigenständige Anwendung ausgeführt oder in mehrere Registerkarten in der AXIS GUI eingebettet werden.
- *PYNGCGUI* ist eine alternative Python-Implementierung von NGCGUI.
- *PyNGCGUI* kann als eigenständige Anwendung laufen oder als Registerkarte (mit einem eigenen Satz von mehreren Unterprogramm-Registerkarten) in jede GUI eingebettet werden, die eine Einbettung der GladeVCP-Anwendungen AXIS, Touchy, Gscreen und GMOCCAPY unterstützt.

NGCGUI oder PyNGCGUI verwenden:

- Für jedes in der INI-Datei angegebene Unterprogramm gibt es Registerkarten (engl. tabs).
- Neue Subroutinen-Registerkarten können mit dem custom tab spontan hinzugefügt werden.
- Jede Registerkarte eines Unterprogramms enthält Eingabefelder für alle Unterprogrammparameter.
- Die Eingabefelder können einen Standardwert und eine Bezeichnung haben, die durch spezielle

Kommentare in der Unterprogrammdatei gekennzeichnet sind.

- Unterprogrammaufrufe können miteinander verkettet werden, um ein mehrschrittiges Programm zu bilden.
- Jede G-Code-Subroutine in einer einzigen Datei, die den NGCGUI-Konventionen entspricht, kann verwendet werden.
- Jedes gcmc-Programm (G-Code-Meta-Compiler), das den NGCGUI-Konventionen für die Kennzeichnung von Variablen entspricht, kann verwendet werden. (Die ausführbare Datei gcmc muss separat installiert werden, siehe: https://www.vagrearg.org/content/gcmc)

NOTE NGCGUI und PyNGCGUI implementieren die gleichen Funktionen und verarbeiten beide .ngc- und .gcmc-Dateien, die einigen NGCGUI-spezifischen Konventionen entsprechen. In diesem Dokument bezieht sich der Begriff "NGCGUI" im Allgemeinen auf beide Anwendungen.

10.6.2. Beispiel-Konfigurationen

Eine Reihe von Demonstrations-Konfigurationen sind in der sim-Verzeichnis der Sample Configurations von der LinuxCNC Konfiguration Picker angeboten befindet. Der Konfigurationspicker befindet sich im Hauptmenü des Systems: Anwendungen > CNC > LinuxCNC

Es sind Beispiele für AXIS, Touchy, Gscreen und GMOCCAPY enthalten. Diese Beispiele demonstrieren sowohl kartesische 3-Achsen-Konfigurationen (wie Fräsmaschinen) als auch Drehbank-Konfigurationen (XZ). Einige Beispiele zeigen die Verwendung einer Pop-up-Tastatur für Touchscreen-Systeme und andere Beispiele demonstrieren die Verwendung von Dateien, die für die Anwendung gcmc (G-code Meta Compiler) erstellt wurden. Die berührungsempfindlichen Beispiele zeigen auch die Einbindung eines GladeVCP-Backplot-Viewers (gremlin_view).

Die einfachste Anwendung ist die folgende:

```
Sample Configurations/sim/axis/ngcgui/ngcgui_simple
```

Ein umfassendes Beispiel für die Kompatibilität von gcmc finden Sie unter:

```
Sample Configurations/sim/axis/ngcgui/ngcgui_gcmc
```

Ein umfassendes Beispiel, das als GladeVCP-App eingebettet ist und gcmc verwendet, finden Sie unter:

```
Sample Configurations/sim/gscreen/ngcgui/pyngcgui_gcmc
```

Die Beispielsimulationskonfigurationen verwenden Bibliotheksdateien, die Beispiel-G-Code-Unterprogrammdateien (.ngc) und G-Code-Meta-Compilerdateien (.gcmc) enthalten:

- *nc_files/ngcgui_lib*
  - *ngcgui.ngc* - Ein leicht verständliches Beispiel mit Unterroutinen
  - *arc1.ngc* - Kreisbogen mit Fräserradiuskompensation

- *arc2.ngc* - Bogen angegeben durch Zentrum, Offset, Breite, Winkel (ruft arc1 auf)
- *backlash.ngc* - Routine zur Messung eines Achsenspiels mit Wählanzeige
- *db25.ngc* - erstellt einen DB25-Plug-Ausschnitt
- *gosper.ngc* - eine Rekursionsdemo (FlowSnake)
- *helix.ngc* - Helix- oder D-Loch-Schneiden
- *helix_rtheta.ngc* - Helix oder D-Loch, die durch Radius und Winkel positioniert sind
- *hole_circle.ngc* - gleichmäßig verteilte Löcher auf einem Kreis
- *ihex.ngc* - internes Sechseck (hexagon)
- *iquad.ngc* - internal quadrilateral
- *ohex.ngc* - äußeres (engl. outside) hexagon
- *oquad.ngc* - äußeres (engl. outside) quadrilateral
- *qpex_mm.ngc* - Demo von qpockets (mm-basiert)
- *qpex.ngc* - Demo von qpockets (zollbasiert)
- *qpocket.ngc* - vierseitige Tasche (lateinisch/englisch quadrilateral pocket)
- *rectangle_probe.ngc* - sondiert einen rechteckigen Bereich
- *simp.ngc* - ein einfaches Beispiel für ein Unterprogramm, das zwei Kreise erzeugt
- *slot.ngc* - Schlitz aus der Verbindung zweier Endpunkte
- *xyz.ngc* - Maschinentrainer, der auf eine Kastenform beschränkt ist
- *Custom* - Erzeugt Benutzer-angepasste Registrierkarten (engl. tabs)
- *ttt* - True Type Tracer, um Texte zu erstellen, die graviert werden sollen

- *nc_files/ngcgui_lib/lathe*
  - ngcgui-lathe' - Beispiel-Unterprogramm für Drehmaschinen
  - *g76base.ngc* - GUI für G76 Gewindebohren
  - g76diam.ngc' - Gewinde nach Haupt- und Nebendurchmesser spezifiziert
  - *id.ngc* - bohrt den Innendurchmesser
  - *od.ngc* - dreht den Außendurchmesser
  - taper-od.ngc' - dreht einen Kegel auf dem Außendurchmesser
  - *Custom* - Erzeugt Benutzer-angepasste Registrierkarten (engl. tabs)
- *nc_files/gcmc_lib*
  - *drill.gcmc* - Löcher im Rechteckmuster bohren
  - square.gcmc' - einfache Demo von variablen Tags für gcmc-Dateien
  - *star.gcmc* - GCMC-Demo zur Veranschaulichung von Funktionen und Arrays
  - *wheels.gcmc* - GCMC Demo komplexer Muster

Um eine Demonstration zu versuchen, wählen Sie eine Sim-Konfiguration und starten Sie das LinuxCNC-

Programm.

Wenn Sie die AXIS GUI verwenden, drücken Sie auf "Notaus" (engl. E-Stop)  und dann auf "Machine Power"  und dann auf "Refefernzierfahrt aller Achsen" (engl. Home All). Wählen Sie eine NGCGUI-Registerkarte, füllen Sie alle leeren Felder mit sinnvollen Werten aus und drücken Sie auf "Feature anlegen" (engl. create feature) und dann auf "Finalize". Drücken Sie abschließend auf die Schaltfläche "Ausführen"  , um die Ausführung zu beobachten. Experimentieren Sie, indem Sie mehrere Merkmale und Merkmale aus verschiedenen Registerkarten erstellen.

Um mehrere Unterprogramme zu erstellen, die in einer einzigen Datei zusammengefasst sind, gehen Sie zu jeder Registerkarte, füllen Sie die Leerstellen aus und drücken Sie "Feature erstellen". Drücken Sie nun auf "Abschließen" und beantworten Sie die Aufforderung zum Erstellen von

Andere GUIs haben eine ähnliche Funktionalität, aber die Schaltflächen und Namen können unterschiedlich sein.

NOTE

Die Demonstrationskonfigurationen erstellen Registerkarten für nur einige der mitgelieferten Beispiele. Jede GUI mit einem angepassten Registrierkarte kann jede der Bibliotheks-Beispiel-Subroutinen oder jede Benutzerdatei öffnen, wenn sie sich im LinuxCNC-Subroutinenpfad befindet.

Um spezielle Tastenbelegungen zu sehen, klicken Sie in eine ngcgui-Registerkarte, um den Fokus zu erhalten, und drücken Sie dann STRG-K.

Die Demonstrationsunterprogramme sollten auf den simulierten Maschinenkonfigurationen laufen, die in der Distribution enthalten sind. Ein Benutzer sollte immer das Verhalten und den Zweck eines Programms verstehen, bevor er es auf einer echten Maschine ausführt.

10.6.3. Bibliothek (engl. library)-Verzeichnisse (engl. locations)

In LinuxCNC-Installationen, die aus .deb-Paketen installiert wurden, verwenden die Simulationskonfigurationen für NGCGUI symbolische Links zu nicht vom Benutzer beschreibbaren LinuxCNC-Bibliotheken für:

- *nc_files/ngcgui_lib* NGCGUI-kompatible Unterdateien
- *nc_files/ngcgui_lib/lathe* NGCGUI-kompatible Drehmaschinen-Unterdateien
- *nc_files/gcmc_lib* NGCGUI-gcmc-kompatible Programme
- *nc_files/ngcgui_lib/utilitysubs* Hilfs-Unterprogramme
- *nc_files/ngcgui_lib/mfiles* Benutzer-M-Dateien

Diese Bibliotheken werden durch INI-Datei-Elemente gefunden, in denen die Suchpfade von LinuxCNC (und NGCGUI) stehen:

```
[RS274NGC]
SUBROUTINE_PATH =
../../nc_files/ngcgui_lib:../../nc_files/gcmc_lib:../../nc_files/ngcgui_lib/utilitysubs
```

```
USER_M_PATH     = ../../nc_files/ngcgui_lib/mfiles
```

NOTE Dabei handelt es sich um lange Zeilen (die nicht über mehrere Zeilen fortgesetzt werden), in denen die in einem Suchfeld verwendeten Verzeichnisse angegeben werden. Die Verzeichnisnamen werden durch Doppelpunkte (:) getrennt. Zwischen den Verzeichnisnamen sollten keine Leerzeichen stehen.

Ein Benutzer kann neue Verzeichnisse für seine eigenen Unterprogramme und M-Dateien erstellen und sie zu den Suchpfaden hinzufügen.

So könnte ein Benutzer beispielsweise Verzeichnisse vom Terminal aus mit den folgenden Befehlen erstellen:

```
mkdir /home/myusername/mysubs
mkdir /home/myusername/mymfiles
```

Und dann die vom System bereitgestellten Dateien in diesen vom Benutzer beschreibbaren Verzeichnisse anlegen oder dorthin kopieren. Ein Benutzer könnte zum Beispiel eine NGCGUI-kompatible Unterdatei namens:

```
/home/myusername/mysubs/example.ngc
```

Um Dateien in neuen Verzeichnissen zu verwenden, muss die INI-Datei bearbeitet werden, um die neuen Unterdateien einzuschließen und den Suchpfad/die Suchpfade zu ergänzen. Für dieses Beispiel:

```
[RS274NGC]
# ...
SUBROUTINE_PATH =
/home/myusername/mysubs:../../nc_files/ngcgui_lib:../../nc_files/gcmc_lib:../../nc_files/
ngcgui_lib/utilitysubs
USER_M_PATH     = /home/myusername/mymfiles:../../nc_files/ngcgui_lib/mfiles

[DISPLAY]
# ...
NGCGUI_SUBFILE = example.ngc
# ...
```

LinuxCNC (und NGCGUI) verwenden die erste gefundene Datei bei der Suche nach Verzeichnissen im Suchpfad. Mit diesem Verhalten können Sie eine ngcgui_lib Unterdatei ersetzen, indem Sie eine Unterdatei mit einem identischen Namen in einem Verzeichnis platzieren, das früher in der Pfadsuche gefunden wird. Weitere Informationen finden Sie im INI-Kapitel des Integrators Manual.

10.6.4. Standalone-Nutzung

Eigenständiges NGCGUI

Zur Verwendung geben Sie in ein Terminal ein:

```
ngcgui --help
Usage:
  ngcgui --help | -?
  ngcgui [Options] -D <nc-Dateien Verzeichnisname>
  ngcgui [Options] -i <LinuxCNC INI Dateiname>
  ngcgui [Options]

Optionen:
  [-S subroutine_file]
  [-p preamble_file]
  [-P postamble_file]
  [-o output_file]
  [-a autosend_file] (automatisches Senden an AXIS) (Voreinstellung:auto.ngc)
  [--noauto] (keine automatische Übertragung an AXIS)
  [-N | --nom2] (kein m2-Terminator (% verwenden))
  [--font [big|small|fontspec]] (Voreinstellung: "Helvetica -10 normal")
  [--horiz|--vert] (Voreinstellung: --horiz)
  [--cwidth comment_width] (Breite des Kommentarfeldes)
  [--vwidth varname_width] (Breite des Feldes varname)
  [--quiet] (weniger Kommentare in der Ausgabedatei)
  [--noiframe] (Voreinstellung: Rahmen zeigt Bild an)
```

NOTE Als eigenständige Anwendung bearbeitet NGCGUI eine einzelne Unterprogrammdatei, die mehrfach aufgerufen werden kann. Mehrere eigenständige NGCGUI-Anwendungen können unabhängig voneinander gestartet werden.

Eigenständiges (engl. standalone) PyNGCGUI

Zur Verwendung geben Sie in ein Terminal ein:

```
pyngcgui --help
Verwendung:
pyngcgui [Optionen] [<sub_filename>]
Optionen, die Werte erfordern:
    [-d | --demo] [0|1|2] (0: DEMO eigenständige Toplevel)
                          (1: DEMO neues Notizbuch einbetten)
                          (2: DEMO innerhalb eines bestehenden Notizbuchs einbetten)
    [-S | --subfile <sub_filename>]
    [-p | --preamble <preamble_filename>]
    [-P | --postamble <postamble_filename>]
    [-i | --ini <inifile_name>]
    [-a | --autofile <auto_filename>]
    [-t | --test <testno>]
    [-K | --keyboardfile <glade_file>] (benutzerdefinierte popupkeyboard glade-Datei
verwenden)
Solo-Optionen:
    [-v | --verbose]
    [-D | --debug]
    [-N | --nom2] (kein m2-Terminator (% verwenden))
    [-n | --noauto] (speichern, aber Ergebnis nicht automatisch senden)
    [-k | --keyboard] (Standard-Popupkeybaord verwenden)
    [-s | --sendtoaxis] (generierte NGC-Datei an Achsen-GUI senden)
Anmerkungen:
```

```
      Ein Satz von Dateien besteht aus einer Präambel, einer Unterdatei und einer
Postambel.
      Die Präambel und Postambel sind optional.
      Ein Satz von Dateien kann über cmdline angegeben werden.
      Mehrere Dateisätze können über eine INI-Datei angegeben werden.
      Wenn --ini NICHT angegeben ist:
         Suche nach einem laufenden LinuxCNC und verwende dessen INI-Datei
```

NOTE Als eigenständige Anwendung kann PyNGCGUI eine INI-Datei (oder eine laufende LinuxCNC-Anwendung) lesen, um Registerkarten für mehrere Unterdateien zu erstellen.

10.6.5. NGCGUI einbetten

NGCGUI in AXIS einbetten

Die folgenden INI-Datei-Elemente gehören in den Abschnitt [DISPLAY]. (Siehe weitere Abschnitte unten für zusätzlich benötigte Elemente)

- *TKPKG = Ngcgui 1.0* - das NGCGUI-Paket
- *TKPKG = Ngcguittt 1.0* - das True Type Tracer-Paket zum Generieren von Text für die Gravur (optional, muss TKPKG = Ngcgui folgen).
- *NGCGUI_FONT = Helvetica -12 normal* - Legt die verwendete Schriftart fest
- NGCGUI_PREAMBLE = in_std.ngc' - Die Präambel-Datei, die am Anfang des Unterprogramms hinzugefügt wird. Bei der Verkettung mehrerer Unterprogramme wird sie nur einmal hinzugefügt.
- *NGCGUI_SUBFILE = simp.ngc* - Erstellt eine Registerkarte aus der benannten Unterroutine.
- *NGCGUI_SUBFILE = ""* - Erzeugt eine benutzerdefinierte Registerkarte
- *#NGCGUI_OPTIONS = opt1 opt2 ...* - NGCGUI-Optionen:
  - *nonew* — Verhindert die Erstellung einer neuen benutzerdefinierten Registerkarte
  - *noremove* — Verbietet das Löschen einer Registerkartenseite
  - *noauto* — Nicht automatisch ausführen (makeFile, dann manuell ausführen)
  - *noiframe* — Kein internes Bild, Bild auf separater oberster Ebene
- *TTT = truetype-tracer* - Name des truetype tracer-Programms (es muss im Benutzer-PATH sein)
- *TTT_PREAMBLE = in_std.ngc* - Optional, gibt den Dateinamen der Präambel an, die für ttt erstellten Unterdateien verwendet wird. (alternativ: mm_std.ngc)

NOTE Die optionalen truetype-tracer-Elemente werden verwendet, um eine NGCGUI-kompatible Registerkarte anzugeben für die Anwendung von truetype-tracer. Die truetype-tracer-Anwendung muss unabhängig installiert werden und sich im Benutzer-PATH befinden.

PyNGCGUI als GladeVCP-Registerkarte in ein GUI einbetten

Die folgenden INI-Datei-Elemente gehören in den Abschnitt [DISPLAY] zur Verwendung mit den grafischen Benutzeroberflächen AXIS, Gscreen oder Touchy. (Weitere benötigte Elemente finden Sie in den folgenden Abschnitten)

EMBED_ Items

- `EMBED_TAB_NAME = PyNGCGUI` - Name, der auf der eingebetteten Registerkarte erscheinen soll
- `EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui` - ruft GladeVCP auf
- `EMBED_TAB_LOCATION = name_of_location` - wo sich die eingebettete Seite befindet

NOTE Der EMBED_TAB_LOCATION-Spezifizierer wird nicht für die AXIS-GUI verwendet. Während PyNGCGUI in AXIS eingebettet werden kann, ist die Integration vollständiger, wenn NGCGUI verwendet wird (mit TKPKG = Ngcgui 1.0). Um die EMBED_TAB_LOCATION für andere GUIs festzulegen, vgl. den Abschnitt zu *DISPLAY* des INI-Konfigurationskapitels.

NOTE Das Truetype Tracer GUI-Frontend ist derzeit nicht für GladeVCP-Anwendungen verfügbar.

Zusätzliche INI-Datei-Elemente, die für NGCGUI oder PyNGCGUI erforderlich sind

Die folgenden INI-Datei-Elemente gehören in den Abschnitt [DISPLAY] für jede GUI, die entweder NGCGUI oder PyNGCGUI einbindet.

- NGCGUI_FONT = Helvetica -12 normal' - gibt den Namen und die Größe der Schriftart an, normal|fett (engl. bold)
- NGCGUI_PREAMBLE = in_std.ngc' - die Präambel-Datei, die den Unterprogrammen vorangestellt wird. Bei der Verkettung mehrerer gemeinsamer Subroutinenaufrufe wird diese Präambel nur einmal hinzugefügt. Für mm-basierte Maschinen verwenden Sie mm_std.ngc
- *NGCGUI_SUBFILE = filename1.ngc* - erstellt eine Registerkarte aus der Unterroutine filename1
- *NGCGUI_SUBFILE = filename2.ngc* - erstellt eine Registerkarte aus der Unterroutine filename2
- *... usw.*
- *NGCGUI_SUBFILE = gcmcname1.gcmc* - erstellt eine Registerkarte aus der Datei gcmcname1
- *NGCGUI_SUBFILE = gcmcname2.gcmc* - erstellt eine Registerkarte aus der Datei gcmcname2
- *... usw.*
- *NGCGUI_SUBFILE = ""* - erstellt eine benutzerdefinierte Registerkarte, die jede Unterroutine im Suchpfad öffnen kann
- *NGCGUI_OPTIONS = opt1 opt2 ...* - NGCGUI-Optionen
  - *nonew* - Erstellen eines neuen benutzerdefinierten Tabs nicht zugelassen
  - *noremove* - das Entfernen von Tab-Seiten nicht zugelassen
  - *noauto* - kein automatisches Senden (makeFile verwenden, dann speichern oder manuell

senden)

- *noiframe* - kein internes Bild, Bilder auf separatem Top-Level-Widget anzeigen
- *nom2* - nicht mit m2 abschließen, sondern %-Terminator verwenden. Diese Option beseitigt alle Nebeneffekte der m2-Terminierung

- GCMC_INCLUDE_PATH = dirname1:dirname2' - sucht Verzeichnisse nach gcmc-Include-Dateien

Dies ist ein Beispiel für die Einbettung von NGCGUI in AXIS. Die Unterprogramme müssen sich in einem Verzeichnis befinden, das durch den [RS274NGC]SUBROUTINE_PATH angegeben ist. Einige Beispielsubroutinen verwenden andere Subroutinen, daher sollten Sie sicherstellen, dass Sie die Abhängigkeiten, falls vorhanden, in einem SUBROUTINE_PATH-Verzeichnis haben. Einige Unterprogramme können benutzerdefinierte M-Dateien verwenden, die sich in einem durch [RS274NGC]USER_M_PATH angegebenen Verzeichnis befinden müssen.

Der G-Code-Meta-Compiler (gcmc) kann Anweisungen wie diese enthalten:

```
include("filename.inc.gcmc");
```

Standardmäßig schließt gcmc das aktuelle Verzeichnis ein, das für LinuxCNC das Verzeichnis ist, das die LinuxCNC INI-Datei enthält. Zusätzliche Verzeichnisse können der gcmc-Suchreihenfolge mit dem Element GCMC_INCLUDE_PATH vorangestellt werden.

Beispiel einer AXIS-GUI-basierten INI

```
[RS274NGC]
# ...
SUBROUTINE_PATH   = ../../nc_files/ngcgui_lib:../../ngcgui_lib/utilitysubs
USER_M_PATH       = ../../nc_files/ngcgui_lib/mfiles

[DISPLAY]
TKPKG = Ngcgui 1.0
TKPKG = Ngcguittt 1.0
# Ngcgui muss vor Ngcguittt stehen

NGCGUI_FONT       = Helvetica -12 normal
# nur Dateinamen angeben, Dateien müssen sich im [RS274NGC]SUBROUTINE_PATH befinden
NGCGUI_PREAMBLE   = in_std.ngc
NGCGUI_SUBFILE    = simp.ngc
NGCGUI_SUBFILE    = xyz.ngc
NGCGUI_SUBFILE    = iquad.ngc
NGCGUI_SUBFILE    = db25.ngc
NGCGUI_SUBFILE    = ihex.ngc
NGCGUI_SUBFILE    = gosper.ngc
# angeben von "" für eine benutzerdefinierte Registerkarte
NGCGUI_SUBFILE    = ""
#NGCGUI_SUBFILE   = "" verwenden, wenn ein Bildrahmen angegeben ist,
# wenn das Öffnen anderer Dateien erforderlich ist
# Bilder werden in ein Fenster der obersten Ebene gestellt
NGCGUI_OPTIONS    =
#NGCGUI_OPTIONS   = opt1 opt2 ...
# opt items:
#   nonew      -- disallow making a new custom tab
#   noremove   -- disallow removing any tab page
```

```
#   noauto     -- no auto send (makeFile, then manually send)
#   noiframe   -- no internal image, image on separate top level
GCMC_INCLUDE_PATH = /home/myname/gcmc_includes

TTT               = truetype-tracer
TTT_PREAMBLE      = in_std.ngc

PROGRAM_PREFIX    = ../../nc_files
```

NOTE Die obige Datei ist keine vollständige AXIS GUI INI — die gezeigten Elemente sind diejenigen, die von NGCGUI verwendet werden. Viele zusätzliche Elemente werden von LinuxCNC erforderlich, um eine vollständige INI-Datei haben.

Truetype Tracer

Ngcgui_ttt bietet Unterstützung für truetype-tracer (v4). Es erstellt eine AXIS-Registerkarte, die es dem Benutzer ermöglicht, eine neue NGCGUI-Registerkarte zu erstellen, nachdem er Text eingegeben und eine Schriftart sowie andere Parameter ausgewählt hat. (Truetype-tracer muss unabhängig installiert werden).

Um ngcgui_ttt in AXIS einzubetten, geben Sie zusätzlich zu den NGCGUI-Elementen die folgenden Elemente an:

```
Element: [DISPLAY]TKPKG = Ngcgui_ttt version_number
Beispiel: [DISPLAY]TKPKG = Ngcgui_ttt 1.0
Hinweis: Obligatorisch, gibt das Laden von ngcgui_ttt in einer AXIS-Registerkarte namens
ttt.
         Muss auf den Eintrag TKPKG = Ngcgui folgen.

Element: [DISPLAY]TTT = pfad_zum_truetype-tracer
Beispiel: [DISPLAY]TTT = truetype-tracer
Note: Optional, wenn nicht angegeben, wird der Pfad /usr/local/bin/truetype-tracer
versucht.
         Angabe mit absolutem Pfadnamen oder als einfacher ausführbarer Befehl.
         In diesem Fall wird die PATH-Umgebung des Benutzers verwendet, um das Programm
zu finden.

Element: [DISPLAY]TTT_PREAMBLE = preamble_filename
Beispiel: [DISPLAY]TTT_PREAMBLE = in_std.ngc
Hinweis: Optional, gibt den Dateinamen der Präambel an, für die von ttt erstellten
Unterdateien.
```

INI-Datei Pfad-Spezifikationen

NGCGUI verwendet den LinuxCNC-Suchpfad, um Dateien zu finden. Der Suchpfad beginnt mit dem Standardverzeichnis, das angegeben wird durch:

```
[DISPLAY]PROGRAM_PREFIX = verzeichnis_name
```

gefolgt von mehreren Verzeichnissen, angegeben durch:

```
[RS274NGC]SUBROUTINE_PATH = Verzeichnis1_Name:Verzeichnis1_Name:Verzeichnis3_Name ...
```

Verzeichnisse

Verzeichnisse (engl. directories) können als absolute Pfade oder relative Pfade angegeben werden.

- Beispiel: `[DISPLAY]PROGRAM_PREFIX = /home/myname/linuxcnc/nc_files`
- Beispiel: `[DISPLAY]PROGRAM_PREFIX = ~/linuxcnc/nc_files`
- Beispiel: `[DISPLAY]PROGRAM_PREFIX = .. /.. /nc_files`

Absolute Pfade

Ein absoluter Pfad, der mit einem "/" beginnt, gibt einen vollständigen Dateisystemstandort an. Ein Pfad, der mit "\~/" beginnt, gibt einen Pfad an, der im Home-Verzeichnis des Benutzers beginnt. Ein Pfad, der mit "~Benutzername/" beginnt, legt einen Pfad fest, der im Home-Verzeichnis des Benutzers beginnt.

Relative Pfade

Relative Pfade basieren auf dem Startverzeichnis, also dem Verzeichnis, das die INI-Datei enthält. Die Verwendung relativer Pfade kann das Verschieben von Konfigurationen erleichtern, erfordert aber ein gutes Verständnis der Linux-Pfadangaben.

- *./d0* ist dasselbe wie d0, z. B. ein Verzeichnis mit dem Namen d0 im Startup-Verzeichnis
- `../d1` bezieht sich auf ein Verzeichnis d1 im übergeordneten Verzeichnis
- `../../d2` verweist auf ein Verzeichnis d2 im übergeordneten Verzeichnis des übergeordneten Verzeichnisses
- `../../../d3` usw.

Mehrere Verzeichnisse können mit [RS274NGC]SUBROUTINE_PATH angegeben werden, indem sie durch Doppelpunkte getrennt werden. Das folgende Beispiel veranschaulicht das Format für mehrere Verzeichnisse und zeigt die Verwendung von relativen und absoluten Pfaden.

Beispiel für mehrere Verzeichnisse:

```
[RS274NGC]SUBROUTINE_PATH =
../../nc_files/ngcgui_lib:../../nc_files/ngcgui_lib/utilitysubs:/tmp/tmpngc
```

Dies ist eine lange Zeile, fahren Sie nicht in mehreren Zeilen fort. Wenn LinuxCNC und/oder NGCGUI nach Dateien suchen, wird die erste Datei, die bei der Suche gefunden wird, verwendet.

LinuxCNC (und NGCGUI) muss in der Lage sein, alle Unterprogramme einschließlich der Hilfsroutinen zu finden, die aus den NGCGUI Unterdateien aufgerufen werden. Es ist zweckmäßig, Utility-Subs in einem separaten Verzeichnis zu platzieren, wie im obigen Beispiel angegeben.

Die Distribution enthält das Verzeichnis ngcgui_lib und Demodateien für Präambeln, Subdateien, Postambeln und Hilfsdateien. Um das Verhalten der Dateien zu ändern, können Sie eine beliebige Datei kopieren und sie an einer früheren Stelle des Suchpfads platzieren. Das erste Verzeichnis, das durchsucht wird, ist [DISPLAY]PROGRAM_PREFIX. Sie können dieses Verzeichnis verwenden, aber es ist besser, eigene Verzeichnisse zu erstellen und sie an den Anfang des [RS274NGC]SUBROUTINE_PATH zu

stellen.

Im folgenden Beispiel werden die Dateien in /home/myname/linuxcnc/mysubs vor den Dateien in ../../nc_files/ngcgui_lib gefunden.

Beispiel für das Hinzufügen eines Benutzerverzeichnisses:

```
[RS274NGC]SUBROUTINE_PATH =
/home/myname/linuxcnc/mysubs:../../nc_files/ngcgui_lib:../../nc_files/ngcgui_lib/utilitys
ubs
```

Neue Benutzer können versehentlich versuchen, Dateien zu verwenden, die nicht so strukturiert sind, dass sie mit den Anforderungen von NGCGUI kompatibel sind. NGCGUI wird wahrscheinlich zahlreiche Fehler melden, wenn die Dateien nicht nach seinen Konventionen kodiert sind. Gute Praxis legt nahe, dass ngcgui-kompatible Unterdateien in ein dafür vorgesehenes Verzeichnis gelegt werden sollten und dass Präambel-, Postambel- und Hilfsdateien in separaten Verzeichnissen liegen sollten, um Versuche, sie als Unterdateien zu verwenden, zu unterbinden. Dateien, die nicht für die Verwendung als Unterdateien vorgesehen sind, können einen speziellen Kommentar enthalten: "(not_a_subfile)", so dass NGCGUI sie automatisch mit einer entsprechenden Meldung zurückweist.

Zusammenfassung der Details der INI-Datei für die Verwendung von NGCGUI

[RS274NGC]SUBROUTINE_PATH = dirname1:dirname2:dirname3 ...

Beispiel: `[RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ngcgui_lib/utilitysubs`
Hinweis: Optional, aber sehr nützlich, um Unterdateien und Utility-Dateien zu organisieren.

[RS274NGC]USER_M_PATH = dirname1:dirname2:dirname3 ...

Beispiel: `[RS274NGC]USER_M_PATH = ../../nc_files/ngcgui_lib/mfiles`
Hinweis: Optional, wird benötigt, um benutzerdefinierte M-files zu finden.

[DISPLAY]EMBED_TAB_NAME = Name, der auf der eingebetteten Registerkarte angezeigt wird

Beispiel: `[DISPLAY]EMBED_TAB_NAME = Pyngcgui`
Hinweis: Die Einträge: `EMBED_TAB_NAME`, `EMBED_TAB_COMMAND`, `EMBED_TAB_LOCATION` definieren eine eingebettete Anwendung für mehrere LinuxCNC-GUIs.

[DISPLAY]EMBED_TAB_COMMAND = Programmname gefolgt von Argumenten

Beispiel: `[DISPLAY]EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui`
Hinweis: Für GladeVCP-Anwendungen siehe das GladeVCP Kapitel.

[DISPLAY]EMBED_TAB_LOCATION = Name_des_Ortes (engl. location)

Beispiel: `[DISPLAY]EMBED_TAB_LOCATION = notebook_main`
Hinweis: Siehe Beispiel-INI-Dateien für mögliche Orte.
Nicht erforderlich für die AXIS GUI.

[DISPLAY]PROGRAM_PREFIX = Verzeichnisname

Beispiel: `[DISPLAY]PROGRAM_PREFIX = ../../nc_files`
Hinweis: Erforderlich, benötig für zahlreiche LinuxCNC-Funktionen.

Es ist das erste Verzeichnis, das bei der Suche nach Dateien verwendet wird.

[DISPLAY]TKPKG = NGCGUI version_number

Beispiel: `[DISPLAY]TKPKG = Ngcgui 1.0`
Hinweis: Nur für AXIS GUI-Einbettung erforderlich.
Spezifiziert das Laden von NGCGUI AXIS Registerkarten.

[DISPLAY]NGCGUI_FONT = Schriftart_deskriptor

Beispiel: [DISPLAY]NGCGUI_FONT = Helvetica -12 normal
Hinweis: Optional, font_descriptor ist ein tcl-kompatibler Font-Spezifikator mit Elementen für fonttype -fontsize fontweight.
Voreinstellung ist: Helvetica -10 normal.
Kleinere Schriftgrößen können für kleine Bildschirme nützlich sein.
Größere Schriftgrößen können für Touchscreen-Anwendungen hilfreich sein.

[ANZEIGE] NGCGUI_SUBFILE = subfile_filename

Beispiel: `[DISPLAY]NGCGUI_SUBFILE = simp.ngc`
Beispiel: `[ANZEIGE]NGCGUI_SUBFILE = square.gcmc`
Beispiel: `[DISPLAY]NGCGUI_SUBFILE = ""` + *Beispiel*: `[DISPLAY]NGCGUI_SUBFILE = ""`
Hinweis: Verwenden Sie ein oder mehrere Elemente, um NGCGUI-kompatible Unterdateien oder gcmc-Programme anzugeben, die beim Start eine Registerkarte benötigen.
Eine "Benutzerdefinierte" Registerkarte wird erstellt, wenn der Dateiname "" lautet.
Ein Benutzer kann eine "Custom"-Registerkarte verwenden, um das Dateisystem zu durchsuchen und Präambel-, Subfile- und Postambel-Dateien zu identifizieren.

[DISPLAY]NGCGUI_PREAMBLE = preamble_filename

Beispiel: `[DISPLAY]NGCGUI_PREAMBLE = in_std.ngc`
Hinweis: Optional, wenn angegeben, wird die Datei einer Unterdatei vorangestellt.
Dateien, die mit "Benutzerdefinierten" Registerkarten erstellt werden, verwenden die mit der Seite angegebene Preamble.

[ANZEIGE] NGCGUI_POSTAMBLE = postamble_filename

Beispiel: `[DISPLAY]NGCGUI_PREAMBLE = in_std.ngc`
Hinweis: Optional, wenn angegeben, wird die Datei einer Unterdatei vorangestellt.
Dateien, die mit "Benutzerdefinierten" Registerkarten erstellt werden, verwenden die mit dieser Seite angegebene Postamble ("Abspann").

[DISPLAY]NGCGUI_OPTIONS = opt1 opt2 ...

Beispiel: [DISPLAY]NGCGUI_OPTIONS = nonew noremove
Hinweis: Mehrere Optionen werden durch Leerzeichen getrennt.
Standardmäßig konfiguriert NGCGUI die Registerkarten derart, dass:
1) ein Benutzer neue Registerkarten erstellen kann;
2) ein Benutzer Registerkarten entfernen kann (mit Ausnahme der letzten verbleibenden Registerkarte);
3) fertiggestellte Dateien automatisch an LinuxCNC gesendet werden;
4) ein Bildrahmen (iframe) wird zur Verfügung gestellt, um ein Bild für die Unterdatei anzuzeigen (falls ein Bild bereitgestellt wird);

5) die an LinuxCNC gesendete NGCGUI Ergebnisdatei wird mit einem M2 beendet (und verursacht M2 Nebeneffekte).

Die Optionen `nonew`, `noremove`, `noauto`, `noiframe`, `nom2` schalten diese Standardverhaltensweisen jeweils aus.

Standardmäßig wird, wenn eine Bilddatei (.png,.gif,jpg,pgm) in demselben Verzeichnis wie die Unterdatei gefunden wird, das Bild im iframe angezeigt. Bei Angabe der Option "noiframe" werden zusätzliche Schaltflächen zur Verfügung gestellt für die Auswahl von Präambel, Subdatei und Postambel sowie zusätzliche Kontrollkästchen. Die Auswahl der Kontrollkästchen sind immer mit speziellen Tasten verfügbar:
`Strg-R` Umschalten von "Werte beim Lesen von Subfiles beibehalten",
`Strg-E` Umschalten "Unterprogramm erweitern",
`Strg-a` Umschalten "Automatisches Senden",
`Strl-k` listet alle Tasten und Funktionen auf.

Wenn `noiframe` angegeben ist und eine Bilddatei gefunden wird, so wird das Bild in einem separaten Fenster angezeigt und alle Funktionen sind auf der Registerkarte verfügbar. Die `NGCGUI_OPTIONS` gelten für alle NGCGUI-Registerkarten mit der Ausnahme, dass die Optionen `nonew`, `noremove` und `noiframe` nicht für durch den Anwender gestaltete (engl. custom) Registerkarten gelten. Verwenden Sie keine "Custom"-Registerkarten, wenn Sie die Möglichkeit des Benutzers einschränken wollen, Unterdateien auszuwählen oder zusätzliche Registerkarten zu erstellen.

[DISPLAY]GCMC_INCLUDE_PATH = dirname1:dirname2:...

Beispiel: `[DISPLAY]GCMC_INCLUDE_PATH = /home/myname/gcmc_includes:/home/myname/gcmc_includes2`
Hinweis: Optional, jedes Verzeichnis wird einbezogen, wenn gcmc aufgerufen wird mit der Option: `--include dirname`.

10.6.6. Dateianforderungen für NGCGUI-Kompatibilität

Anforderungen an eine G-code-Unterroutine (.ngc) in einer Datei

Eine NGCGUI-kompatible Unterdatei enthält eine einzelne Unterprogrammdefinition. Der Name der Subroutine muss derselbe sein wie der Dateiname (ohne das Suffix .ngc). LinuxCNC unterstützt benannte oder nummerierte Subroutinen, aber nur benannte Subroutinen sind mit NGCGUI kompatibel. Für weitere Informationen siehe das Kapitel O-Codes.

Die erste unkommentierte Zeile sollte eine `sub`-Anweisung sein.
Die letzte unkommentierte Zeile sollte eine `endsub`-Anweisung sein.

examp.ngc:

```
(info: info_text_zu_erscheinen_oben_auf_der_Tabellenseite)
; Kommentarzeile beginnend mit Semikolon
( Kommentarzeile mit Klammern)
o<examp> sub
  KÖRPER_DER_UNTERROUTINE
```

```
o<examp> endsub
; Kommentarzeile beginnend mit Semikolon
( Kommentarzeile mit Klammern)
```

Der Hauptteil (Körper, engl. body) des Unterprogramms sollte mit einer Reihe von Anweisungen beginnen, die lokale benannte Parameter für jeden für den Unterprogrammaufruf erwarteten Positionsparameter definieren. Diese Definitionen müssen fortlaufend sein, beginnend mit #1 und endend mit der zuletzt verwendeten Parameternummer. Für jeden dieser Parameter müssen Definitionen angegeben werden (keine Auslassungen).

Nummerierung der Parameter

```
#<xparm> = #1
#<yparm> = #2
#<zparm> = #3
```

LinuxCNC betrachtet alle nummerierten Parameter im Bereich #1 bis #30 als Aufrufparameter, so dass NGCGUI Eingabefelder für jedes Auftreten von Parametern in diesem Bereich zur Verfügung stellt. Es ist eine gute Praxis, um die Verwendung von nummerierten Parametern #1 bis #30 überall sonst in der Subroutine zu vermeiden. Die Verwendung lokaler, benannter Parameter wird für alle internen Variablen empfohlen.

Jede definierende Anweisung kann optional einen speziellen Kommentar und einen Standardwert für den Parameter enthalten.

Ausdruck/Anweisung (engl. statement) Prototyp

```
#<vname> = #n (=Standard_Wert)
oder
#<vname> = #n (kommentar_text)
oder
#<vname> = #n (=Standardwert_Kommentar_text)
```

Beispiele für Parameter

```
#<xparm> = #1 (=0.0)
#<yparm> = #2 (Ystart)
#<zparm> = #3 (=0.0 Z Start Einstellung)
```

Wenn ein default_value angegeben ist, wird dieser beim Start in das Eingabefeld für den Parameter eingetragen. Wenn comment_text angegeben ist, wird dieser anstelle des Parameternamens zur Identifizierung der Eingabe verwendet.

Globale benannte Parameter

Hinweise zu globalen benannten Parametern und NGCGUI:

(globale benannte Parameter haben einen führenden Unterstrich im Namen, wie #<_irgendeinglobalername>)

Wie in vielen Programmiersprachen ist die Verwendung von globalen Parametern mächtig, kann aber oft zu unerwarteten Konsequenzen führen. In LinuxCNC werden bestehende globale benannte

Parameter bei der Ausführung von Unterprogrammen gültig sein und Unterprogramme können globale benannte Parameter ändern oder erstellen.

Von der Übergabe von Informationen an Unterprogramme unter Verwendung globaler benannter Parameter wird abgeraten, da eine solche Verwendung die Einrichtung und Pflege eines genau definierten globalen Kontexts erfordert, der schwer zu pflegen ist. Die Verwendung der nummerierten Parameter Nr. 1 bis Nr. 30 als Unterprogramm-Eingaben sollte ausreichen, um eine breite Palette von Design-Anforderungen zu erfüllen.

NGCGUI unterstützt einige globale benannte Eingabeparameter, aber deren Verwendung ist veraltet und hier nicht dokumentiert.

Während von global benannten Eingabeparametern abgeraten wird, müssen LinuxCNC-Subroutinen global benannte Parameter für die Rückgabe von Ergebnissen verwenden. NGCGUI-kompatible Unterdateien sind auf die Verwendung in der Benutzeroberfläche ausgerichtet. Daher sind Rückgabewerte keine übliche Anforderung. Allerdings ist NGCGUI als Testwerkzeug für Subroutinen nützlich, die global benannte Parameter zurückgeben, und es ist üblich, dass NGCGUI-kompatible Subdateien Utility-Subroutinen aufrufen, die Ergebnisse mit global benannten Parametern zurückgeben.

Um diese Verwendungen zu unterstützen, ignoriert NGCGUI globale benannte Parameter, die einen Doppelpunkt (:) in ihrem Namen enthalten. Die Verwendung des Doppelpunkts (:) im Namen verhindert, dass NGCGUI Eingabefelder für diese Parameter erstellt.

Beispiel für globale benannte Parameter

```
o<examp> sub
...
#<_examp:result> = #5410 (liefert den aktuellen Werkzeugdurchmesser)
...
o<helper> call [#<x1>] [#<x2>] (Aufruf einer Subroutine)
#<xresult> = #<_helper:answer> (lokalisiert sofort das globale Ergebnis des Helfers)
#<_helper:answer> = 0.0 (löscht den globalen benannten Parameter, der von der Subroutine
verwendet wird)
...
o<examp> endsub
```

Im obigen Beispiel befindet sich das Unterprogramm in einer separaten Datei namens helper.ngc. Die helper-Routine liefert ein Ergebnis in einem globalen benannten Parameter namens #<_helper:answer.

Aus Gründen der guten Praxis lokalisiert die aufrufende Teildatei das Ergebnis sofort für die Verwendung an anderer Stelle in der Teildatei. Es wird der globale benannte Parameter genullt, der für die Rückgabe des Ergebnisses verwendet wird, um seine unbeabsichtigte Verwendung an anderer Stelle im globalen Kontext zu verhindern. (Ein Nullifizierungswert von 0,0 ist nicht immer eine gute Wahl).

NGCGUI unterstützt die Erstellung und Verkettung von mehreren Features für ein Subfile und für mehrere Subfiles. Es ist manchmal nützlich, die Reihenfolge der Unterdateien zur Laufzeit zu bestimmen, daher fügt NGCGUI einen speziellen globalen Parameter ein, der in Unterprogrammen getestet werden kann. Der Parameter heißt #<_feature:>. Sein Wert beginnt mit dem Wert 0 und wird für jedes hinzugefügte Feature inkrementiert.

Zusatzfunktionen

Ein spezieller *info*-Kommentar kann überall in einer NGCGUI-kompatiblen Unterdatei eingefügt werden. Das Format ist:

```
(info: info_text)
```

Der info_text wird im oberen Bereich der Registerkarte NGCGUI in AXIS angezeigt.

Dateien, die nicht für die Verwendung als Unterdateien vorgesehen sind, können einen speziellen Kommentar enthalten, so dass NGCGUI sie automatisch mit einer entsprechenden Meldung zurückweist.

```
(not_a_subfile)
```

Eine optionale Bilddatei (.png,.gif,.jpg,.pgm) kann eine Unterdatei begleiten. Die Bilddatei kann zur Verdeutlichung der von der Teildatei verwendeten Parameter beitragen. Die Bilddatei sollte sich im gleichen Verzeichnis wie die Unterdatei befinden und den gleichen Namen mit einem entsprechenden Bildsuffix haben, z.B. könnte die Unterdatei example.ngc von einer Bilddatei examp.png begleitet werden. NGCGUI versucht, große Bilder durch Subsampling auf eine Größe mit einer maximalen Breite von 320 und einer maximalen Höhe von 240 Pixeln zu verkleinern.

Keine der Konventionen, die für die Herstellung einer NGCGUI-kompatiblen Subdatei erforderlich sind, schließen ihre Verwendung als allgemeine Subroutinendatei für LinuxCNC aus.

Die LinuxCNC-Distribution enthält eine Bibliothek (ngcgui_lib Verzeichnis), die sowohl Beispiel NGCGUI-kompatiblen Subdateien und Utility-Dateien, um die Funktionen von LinuxCNC Subroutinen und NGCGUI Verwendung zu veranschaulichen enthält. Eine weitere Bibliothek (gcmc_lib) bietet Beispiele für Unterprogrammdateien für den G-Code-Meta-Compiler (gcmc).

Weitere benutzerdefinierte Subroutinen finden Sie im Forum im Abschnitt zu Subroutinen.

Gcode-Meta-Compiler-Dateianforderungen (.gcmc)

Dateien für den Gcode-Meta-Compiler (gcmc) werden von NGCGUI gelesen und es werden Eingabefelder für die in der Datei markierten Variablen erstellt. Wenn ein Feature für die Datei fertiggestellt ist, übergibt NGCGUI die Datei als Eingabe an den gcmc-Compiler und, wenn die Kompilierung erfolgreich ist, wird die resultierende G-Code-Datei an LinuxCNC zur Ausführung gesendet. Die resultierende Datei wird als Single-File-Subroutine formatiert; .gcmc-Dateien und .ngc-Dateien können von NGCGUI gemischt werden.

Die Variablen, die für die Aufnahme in NGCGUI identifiziert wurden, werden mit Zeilen markiert, die dem gcmc-Compiler als Kommentare erscheinen.

Formate für variable Tags

```
//ngcgui: varname1 =
//ngcgui: varname2 = value2
//ngcgui: varname3 = value3, label3;
```

Beispiele für Variablen-Tags

```
//ngcgui: zsafe =
//ngcgui: feedrate = 10
//ngcgui: xl = 0, x limit
```

In diesen Beispielen hat das Eingabefeld für varname1 keinen Standardwert, das Eingabefeld für varname2 hat den Standardwert 2 und das Eingabefeld für varname 3 hat den Standardwert 3 und die Bezeichnung label3 (statt varname3). Die Standardwerte müssen Zahlen sein.

Um die Änderung gültiger Zeilen in einer gcmc-Datei zu erleichtern, werden alternative Tag-Zeilenformate akzeptiert. Die alternativen Formate ignorieren abschließende Semikolons (;) und abschließende Kommentarzeichen (//). Mit dieser Bestimmung ist es oft möglich, einfach das //ngcgui: Tag zu bestehenden Zeilen in einer .gcmc-Datei hinzuzufügen.

Alternative Variablen-Tag-Formate

```
//ngcgui: varname2 = value2;
//ngcgui: varname3 = value3; //, label3;
```

Beispiele für alternative Variablen-Tags

```
//ngcgui: feedrate = 10;
//ngcgui: xl = 0; //, x limit
```

Eine Info-Zeile, die oben auf einer Registerkarte erscheint, kann optional mit einer Zeile mit der Kennzeichnung als:

Info-Tag

```
//ngcgui: info: text_to_appear_at_top_of_tab_page
```

Falls erforderlich, können Optionen mit einem Zeilen-Tag an den gcmc-Compiler übergeben werden:

Option line tag format

```
//ngcgui: -option_name [ [=] option_value]
```

Beispiele für Options-Zeilen-Tags

```
//ngcgui: -I
//ngcgui: --imperial
//ngcgui: --precision 5
//ngcgui: --precision=6
```

Die Optionen für gcmc sind mit dem Terminalbefehl verfügbar:

```
gcmc --help
```

Ein gcmc-Programm verwendet standardmäßig den metrischen Modus. Mit der Option setting kann der Modus auf Zoll eingestellt werden:

```
//ngcgui: --imperial
```

Eine eventuell verwendete Präambel-Datei kann einen Modus (g20 oder g21) festlegen, der mit dem von einer gcmc-Datei verwendeten Modus kollidiert. Um sicherzustellen, dass der gcmc-Programmmodus in Kraft ist, fügen Sie die folgende Anweisung in die .gcmc-Datei ein:

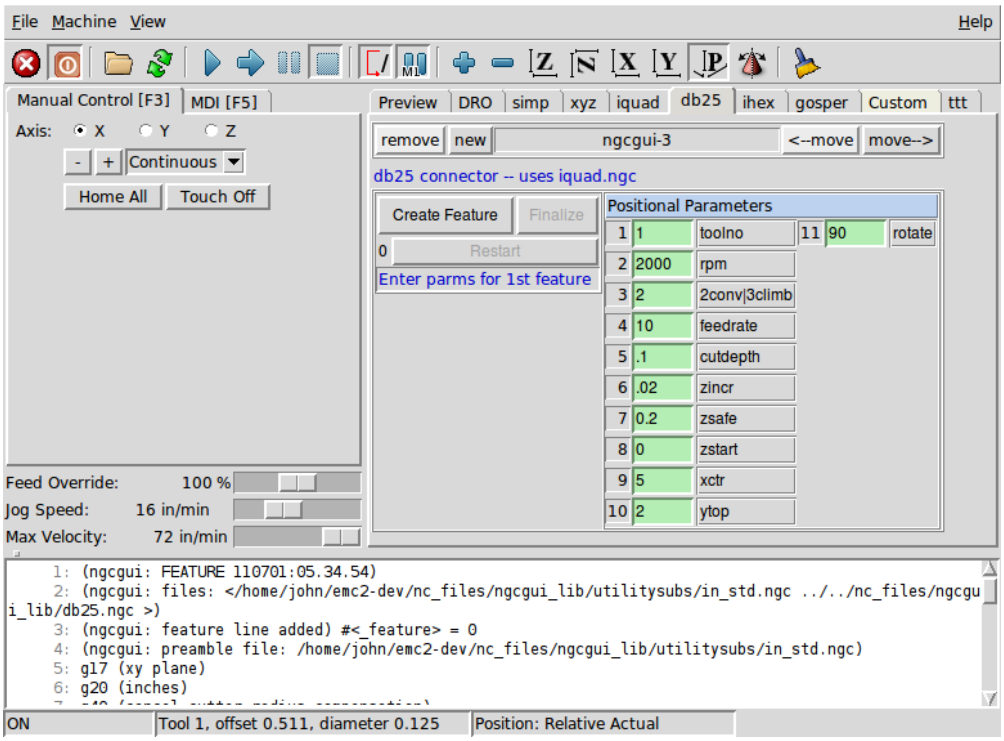
```
include("ensure_mode.gcmc")
```

und geben Sie den richtigen Pfad für gcmc include_files in der INI-Datei an, zum Beispiel:

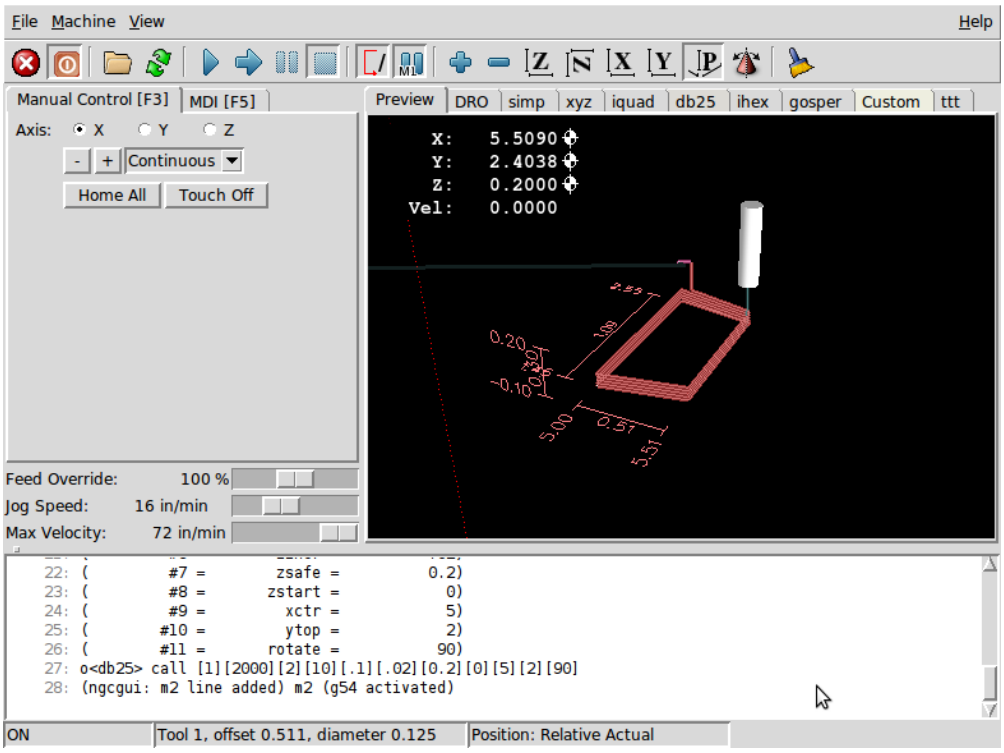
```
[DISPLAY]
GCMC_INCLUDE_PATH = ../../nc_files/gcmc_lib
```

10.6.7. DB25 Beispiel

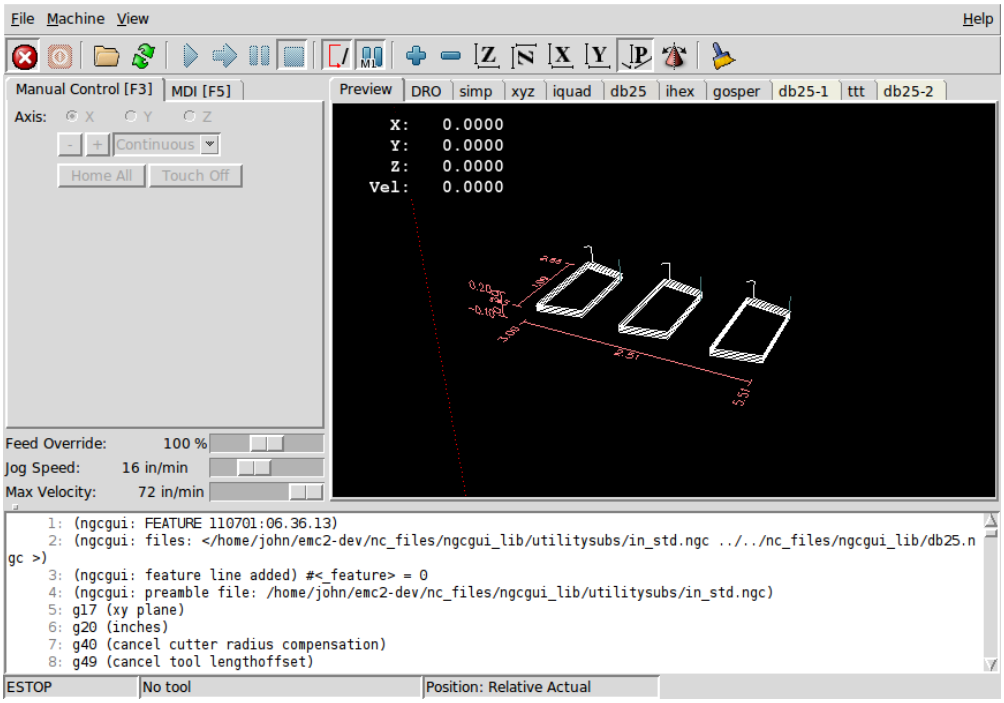
Im Folgenden wird eine DB25 Unterroutine gezeigt. Auf dem ersten Foto sehen Sie, wo Sie die Lücken für jede Variable ausfüllen.



Dieses Foto zeigt den Backplot der DB25-Subroutine.



Dieses Foto zeigt die Verwendung der neuen Schaltfläche und der benutzerdefinierten Registerkarte zur Erstellung von drei DB25-Ausschnitten in einem Programm.



10.6.8. Erstellen eines Unterprogramms

- Um ein Unterprogramm für die Verwendung mit NGCGUI zu erstellen, müssen der Dateiname und der Name des Unterprogramms identisch sein.
- Die Datei muss sich in dem Unterverzeichnis befinden, auf das in der INI-Datei verwiesen wird.
- In der ersten Zeile kann ein Kommentar des Typs `info:` stehen
- Das Unterprogramm muss von den Tags `sub` und `endsub` umgeben sein.

- Die verwendeten Variablen müssen nummerierte Variablen sein und dürfen keine Nummer überspringen.
- Kommentare und Voreinstellungen können enthalten sein.

Unterprogramm-Skelett Beispiel

```
(info: simp -- simple exemple de sous-programme -- Ctrl-U pour éditer)
o<simp> sub
  #<ra> = #1 (=.6 Rayon A) ;Beispiel für einen Parameter mit einem Kommentar
  #<radius_b> = #2 (=0.4) ;Beispiel für einen Parameter ohne Kommentar
  #<feedrate> = #3 (Feedrate) ;Beispiel für einen Parameter ohne Voreinstellung
  g0x0y0z1
  g3 i#<ra> f#<feedrate>
  g3 i[0-#<Radius_b>]
o<simp> endsub
```

10.7. TkLinuxCNC GUI

10.7.1. Einführung

TkLinuxCNC ist eines der ersten grafischen Front-Ends für LinuxCNC. Es ist in Tcl geschrieben und verwendet das Tk-Toolkit für die Anzeige. Es ist in Tcl geschrieben und daher sehr portabel (es läuft auf einer Vielzahl von Plattformen). Ein separates Backplot-Fenster kann wie abgebildet angezeigt werden.

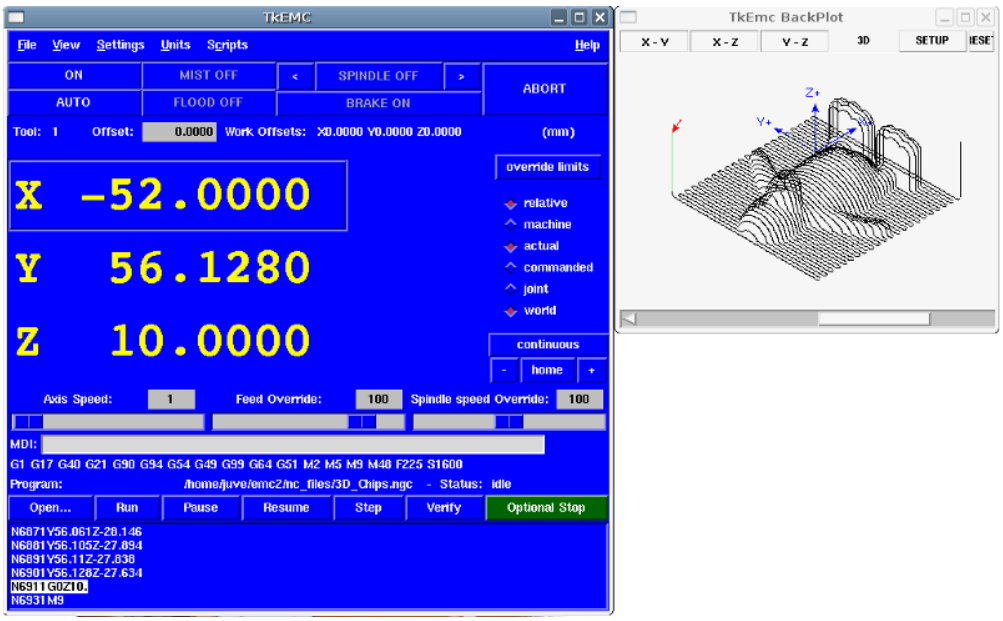


Figure 187. TkLinuxCNC-Fenster

10.7.2. Erste Schritte

Um TkLinuxCNC als Front-End für LinuxCNC wählen, bearbeiten Sie die INI-Datei. Im Abschnitt *[DISPLAY]* ändern Sie die *DISPLAY* Zeile zu lesen

```
DISPLAY = tklinuxcnc
```

Dann starten Sie LinuxCNC und wählen Sie diese INI-Datei. Die Beispielkonfiguration *sim/tklinuxcnc/tklinuxcnc.ini* ist bereits konfiguriert, um TkLinuxCNC als Front-End zu verwenden.

Nach dem Start von LinuxCNC wird das Fenster TKLinuxCNC geöffnet.

Eine typische Sitzung mit TkLinuxCNC

1. Starten Sie LinuxCNC und wählen Sie eine Konfigurationsdatei.
2. Beheben Sie den Notaus (engl. *E-STOP*)-Zustand und schalten Sie die Maschine ein (indem Sie F1 und dann F2 drücken).
3. *Referenzfahrt* jeder Achse.
4. Laden Sie die zu fräsende Datei.
5. Legen Sie das zu fräsende Material auf den Tisch.
6. Stellen Sie die richtigen Versätze für jede Achse ein, indem Sie joggen und entweder erneut referenzieren oder mit der rechten Maustaste auf einen Achsennamen klicken und einen Versatzwert eingeben. Fußnote:[Für einige dieser Aktionen kann es notwendig sein, den Modus zu ändern, in dem LinuxCNC gerade läuft.]
7. Führen Sie das Programm aus.
8. Um dieselbe Feile erneut zu fräsen, kehren Sie zu Schritt 6 zurück. Um eine andere Datei zu fräsen, kehren Sie zu Schritt 4 zurück. Wenn Sie fertig sind, beenden Sie LinuxCNC.

10.7.3. Elemente des TkLinuxCNC-Fensters

Das TkLinuxCNC-Fenster enthält die folgenden Elemente:

- Eine Menüleiste, über die Sie verschiedene Aktionen ausführen können.
- Eine Reihe von Tasten, mit denen Sie den aktuellen Arbeitsmodus, die Start-/Stoppspindel und andere relevante E / A ändern können
- Statusleiste für verschiedene Offset-bezogene Anzeigen
- Koordinatenanzeigebereich
- Eine Reihe von Schiebereglern zur Steuerung von *Jogginggeschwindigkeit*, *Vorschub-Override* und *Spindeldrehzahl-Override*, mit denen Sie diese Einstellungen erhöhen oder verringern können
- Textfeld für die manuelle Dateneingabe *MDI*
- Statusleiste mit aktiven G-Codes, M-Codes, F- und S-Wörtern
- Schaltflächen für Interpreter
- Ein Textfeld, das die G-Code der geladenen Datei anzeigt

Die wichtigsten Buttons

Von links nach rechts lauten die Buttons:

- Maschinenaktivierung: *ESTOP* > *ESTOP RESET* > *ON*

- Kühlnebel ein-/ausschalten
- Spindeldrehzahl verringern
- Spindeldrehrichtung einstellen *SPINDEL AUS* > *SPINDEL VORWÄRTS* . *SPINDEL RÜCKWÄRTS*
- Spindeldrehzahl erhöhen
- Abbrechen

dann in der zweiten Zeile:

- Betriebsart: *MANUAL* > *MDI* > *AUTO*
- Flutkühlmittel ein-/ausschalten
- Spindelbremse ein-/ausschalten

Statusleiste der Offset-Anzeige

Die Statusleiste der Versatzanzeige zeigt das aktuell ausgewählte Werkzeug (ausgewählt mit Txx M6), den Werkzeuglängenversatz (falls aktiv) und die Arbeitsversätze (eingestellt durch Rechtsklick auf die Koordinaten) an.

Koordinatenanzeigebereich

Der Hauptteil der Anzeige zeigt die aktuelle Position des Werkzeugs an. Die Farbe der Positionsanzeige hängt vom Zustand der Achse ab. Wenn die Achse nicht referenziert ist, wird die Achse in gelber Schrift angezeigt. Sobald sie referenziert ist, wird sie in grüner Schrift angezeigt. Wenn es einen Fehler mit der aktuellen Achse TkLinuxCNC werden rote Buchstaben verwendet, um anzuzeigen, dass. (zum Beispiel, wenn eine Hardware-Endschalter ausgelöst wird).

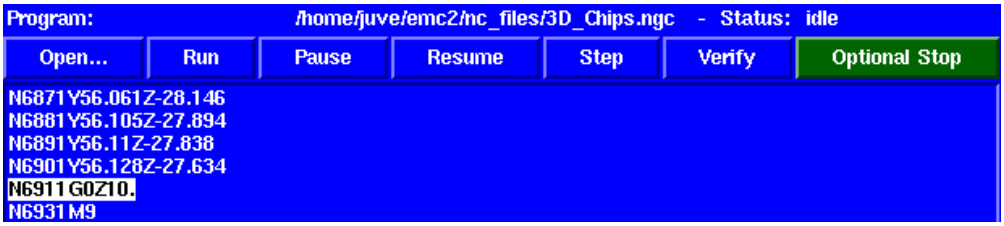
Um diese Zahlen richtig zu interpretieren, beachten Sie die Optionsfelder auf der rechten Seite. Ist die Position "Maschine", dann ist die angezeigte Zahl im Maschinenkoordinatensystem. Bei der Option "Relativ" wird die Zahl im Offset-Koordinatensystem angezeigt. Weiter unten können Sie zwischen "actual" und "commanded" wählen. Ist" bezieht sich auf die Rückmeldung von den Messgeräten (wenn Sie eine Servomaschine haben), und "Soll" bezieht sich auf den Positionsbefehl, der an die Motoren gesendet wird. Diese Werte können sich aus verschiedenen Gründen unterscheiden: Schleppfehler, Totzone, Encoderauflösung oder Schrittweite. Wenn Sie beispielsweise eine Bewegung mit X 0,0033 auf Ihrer Fräsmaschine befehlen, aber ein Schritt Ihres Schrittmotors 0,00125 beträgt, dann ist die *befohlene* Position 0,0033, aber die *tatsächliche* Position ist 0,0025 (2 Schritte) oder 0,00375 (3 Schritte).

Mit einer weiteren Reihe von Optionsfeldern können Sie zwischen "Gelenk-" und "Weltansicht" wählen. Diese sind bei normalen Maschinen (z. B. triviale Kinematik) wenig sinnvoll, helfen aber bei Maschinen mit nicht-trivialer Kinematik wie Robotern oder Stewart-Plattformen. (Sie können mehr über Kinematik im Integrator-Handbuch lesen).

Backplot

Wenn sich die Maschine bewegt, hinterlässt sie eine Spur, den so genannten Backplot. Sie können das Backplot-Fenster starten, indem Sie Ansicht→Backplot wählen.

TkLinuxCNC Interpreter / Automatische Programmsteuerung



Bedientasten

Die Buttons im unteren Teil von TkLinuxCNC werden verwendet, um die Ausführung eines Programms zu steuern:

+ * *Öffnen* (engl. open), um ein Programm zu laden, * *Überprüfen* (engl. verify) um es auf Fehler zu überprüfen, * *Ausführen* (engl. run), um den eigentlichen Schneidevorgang zu starten, * *Pause*, um es während des Laufens anzuhalten, * *Fortsetzen* (engl. resume), um ein bereits angehaltenes Programm wieder aufzunehmen, * *Schritt* (engl. step), um eine Zeile im Programm voranzubringen und * *Optional Stop* zum Umschalten des optionalen Stop-Schalters (wenn die Schaltfläche grün ist, wird die Programmausführung bei jedem M1-Ereignis angehalten).

Anzeigebereich des Textprogramms

Wenn das Programm läuft, wird die Zeile, die gerade ausgeführt wird, weiß hervorgehoben. Die Textanzeige scrollt automatisch, um die aktuelle Zeile anzuzeigen.

Manuelle Steuerung

Steuerung mit der Tastatur

TkLinuxCNC ermöglicht es Ihnen, die Maschine manuell zu bewegen. Diese Aktion wird als "Jogging" bekannt. Wählen Sie zunächst die zu bewegende Achse aus, indem Sie sie anklicken. Klicken Sie dann auf die Schaltfläche "+" oder "-" und halten Sie sie gedrückt, je nach der gewünschten Bewegungsrichtung. Die ersten vier Achsen können auch mit den Pfeiltasten der Tastatur (X und Y), den Tasten PAGE UP und PAGE DOWN (Z) und den Tasten *[* und *]* (A/4) bewegt werden.

+ Wenn Sie *Kontinuierlich* (kontinuierlich) auswählen, wird die Bewegung so lange fortgesetzt, wie die Schaltfläche oder Taste gedrückt wird. Wird ein anderer Wert gewählt, bewegt sich die Maschine jedes Mal, wenn die Schaltfläche angeklickt oder die Taste gedrückt wird, genau um die angezeigte Strecke. Die verfügbaren Werte sind:

+

```
1.0000, 0.1000, 0.0100, 0.0010, 0.0001
```

+ Durch Drücken von "Home" oder der HOME-Taste wird die gewählte Achse in die Grundstellung gebracht. Abhängig von Ihrer Konfiguration kann dies nur den Achsenwert auf die absolute Position 0.0 setzen, oder die Maschine durch Verwendung von *Home-Schaltern* zu einer bestimmten Home-Position fahren lassen. Siehe das Kapitel zur Referenzfahrt für weitere Informationen.

+ Durch Drücken der Taste "Grenzen überschreiben" wird der Maschine vorübergehend erlaubt, außerhalb der in der INI-Datei definierten Grenzen zu verfahren. (Hinweis: Wenn "Grenzen überschreiten" aktiv ist, wird die Schaltfläche in roter Farbe angezeigt).

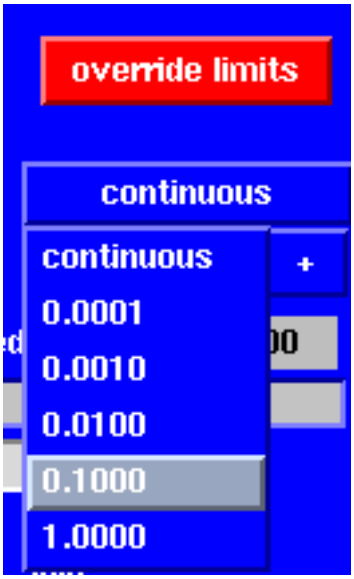


Figure 188. Beispiel für TkLinuxCNC Override Limits & Jogging Increments

Die Spindel-Gruppe

Mit der Taste in der ersten Reihe wird die Drehrichtung der Spindel ausgewählt: Gegen den Uhrzeigersinn, Angehalten, Im Uhrzeigersinn. Mit den Tasten daneben kann der Benutzer die Drehgeschwindigkeit erhöhen oder verringern. Mit der Taste in der zweiten Reihe kann die Spindelbremse aktiviert oder deaktiviert werden. Je nach Maschinenkonfiguration haben möglicherweise nicht alle Elemente in dieser Gruppe eine Wirkung.

Die Kühlmittelgruppe

Mit den beiden Schaltflächen können die Kühlmittel "Nebel" und "Flut" ein- und ausgeschaltet werden. Je nach Konfiguration Ihres Geräts werden möglicherweise nicht alle Elemente in dieser Gruppe angezeigt.

Code-Eingabe

Die manuelle Dateneingabe (auch MDI genannt) ermöglicht die manuelle Eingabe von G-Code-Programmen, eine Zeile nach der anderen. Wenn das Gerät nicht eingeschaltet und nicht auf den MDI-Modus eingestellt ist, sind die Steuerelemente für die Codeeingabe nicht verfügbar.



Hier können Sie einen G-Code-Befehl eingeben, der ausgeführt werden soll. Führen Sie den Befehl aus, indem Sie Enter drücken.

Aktive G-Codes

Hier werden die "Modalcodes" angezeigt, die im Interpreter aktiv sind. Zum Beispiel zeigt *G54* an, dass der *G54 Offset* auf alle eingegebenen Koordinaten angewendet wird.

Jog-Geschwindigkeit

Durch Verschieben dieses Schiebereglers kann die Geschwindigkeit der Jogs geändert werden. Die Zahlen oben beziehen sich auf Achseneinheiten/Sekunde. Das Textfeld mit der Zahl ist anklickbar. Wenn Sie darauf klicken, erscheint ein Popup-Fenster, in das Sie die Zahl eingeben können.

Vorschub Neufestsetzung (engl. override)

Durch Verschieben dieses Schiebereglers kann der programmierte Vorschub geändert werden. Wenn zum Beispiel ein Programm "F60" verlangt und der Schieberegler auf 120% eingestellt ist, dann ist der resultierende Vorschub 72. Das Textfeld mit der Zahl ist anklickbar. Nach dem Anklicken erscheint ein Popup-Fenster, in das eine Zahl eingegeben werden kann.

Spindeldrehzahl Override

Der Schieberegler für die Spindeldrehzahlübersteuerung funktioniert genau wie der Schieberegler für die Vorschubübersteuerung, steuert aber die Spindeldrehzahl. Wenn ein Programm S500 (Spindeldrehzahl 500 U/min) anfordert und der Schieberegler auf 80% eingestellt ist, beträgt die resultierende Spindeldrehzahl 400 U/min. Dieser Schieberegler hat einen Mindest- und einen Höchstwert, die in der INI-Datei definiert sind. Wenn diese fehlen, bleibt der Schieberegler bei 100% stehen. Das Textfeld mit der Zahl ist anklickbar. Sobald es angeklickt wird, erscheint ein Popup-Fenster, in das eine Zahl eingegeben werden kann.

10.7.4. Tastatursteuerung

Fast alle Aktionen in TkLinuxCNC können über die Tastatur ausgeführt werden. Viele der Tastenkombinationen sind nicht verfügbar, wenn im MDI-Modus.

Die am häufigsten verwendeten Tastaturkürzel sind in der folgenden Tabelle aufgeführt.

Table 60. Häufigste Tastaturkürzel

Tastenkombination	Ergriffene Maßnahmen
F1	Notaus ein-/ausschalten
F2	Maschine ein-/ausschalten
`, 1 .. 9, 0	Vorschub-Override von 0% bis 100% einstellen
X, `	Erste Achse aktivieren
Y, 1	Zweite Achse aktivieren
Z, 2	Dritte Achse aktivieren
A, 3	Vierte Achse aktivieren
Pos1	Aktive Achse zu Referenzpunkt (engl. home) schicken

Tastenkombination	Ergriffene Maßnahmen
Links, Rechts	Erste Achse joggen
Hoch, Runter	Zweite Achse joggen
Bild Hoch, Bild Runter (engl. Pg Up, Pg Dn)	Joggen der dritten Achse
[,]	Vierte Achse joggen
Esc	Ausführung stoppen

10.8. QtPlasmaC

10.8.1. Präambel

Sofern nicht anders angegeben, geht diese Anleitung davon aus, dass der Benutzer die neueste Version von QtPlasmaC verwendet. Die Versionsgeschichte kann unter link eingesehen werden, wo die letzte verfügbare Version angezeigt wird. Die installierte QtPlasmaC Version wird in der Titelleiste angezeigt. Siehe Update QtPlasmaC für Informationen zur Aktualisierung von QtPlasmaC.

10.8.2. Lizenz

QtPlasmaC und die gesamte zugehörige Software sind unter der GPLv2 veröffentlicht.

10.8.3. Einführung

Die Version des Entwicklungs-Branch von QtPlasmaC ist eine grafische Benutzeroberfläche für das Plasmaschneiden, welche die plasmac-Komponente zur Steuerung eines Plasmatisches von LinuxCNC v2.10 oder höher unter Verwendung von Debian Bullseye oder einer ähnlichen Distribution verwendet.

Die QtPlasmaC-GUI unterstützt bis zu fünf Achsen und nutzt die QtVCP-Infrastruktur, die mit LinuxCNC bereitgestellt wird.

Das Standardthema basiert auf einem Entwurf des Benutzers "pinder" aus dem LinuxCNC Forum und die Farben können vom Benutzer geändert werden.

Die Entwicklungs-Version der QtPlasmaC GUI läuft auf jeder Hardware, die von derjeweiligen master-branch Version von LinuxCNC (v2.10) unterstützt wird, vorausgesetzt, es gibt genügend Hardware-I/O-Pins, um die Anforderungen einer Plasmakonfiguration zu erfüllen.

Es sind drei Formate verfügbar:

- 16:9 mit einer minimalen Auflösung von 1366 x 768
- 9:16 mit einer minimalen Auflösung von 768 x 1366
- 4:3 mit einer Mindestauflösung von 1024 x 768

Nachfolgend finden Sie einige Screenshot-Beispiele von QtPlasmaC:

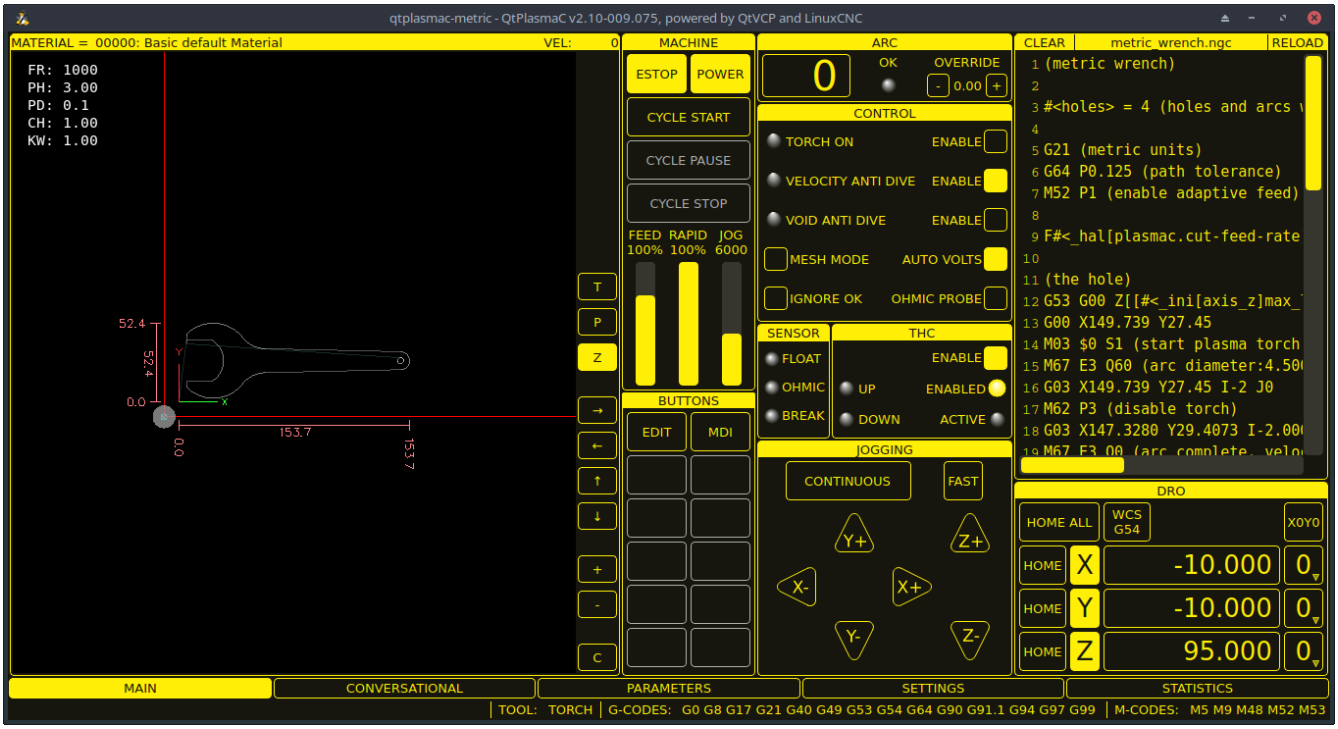


Figure 189. **16:9**

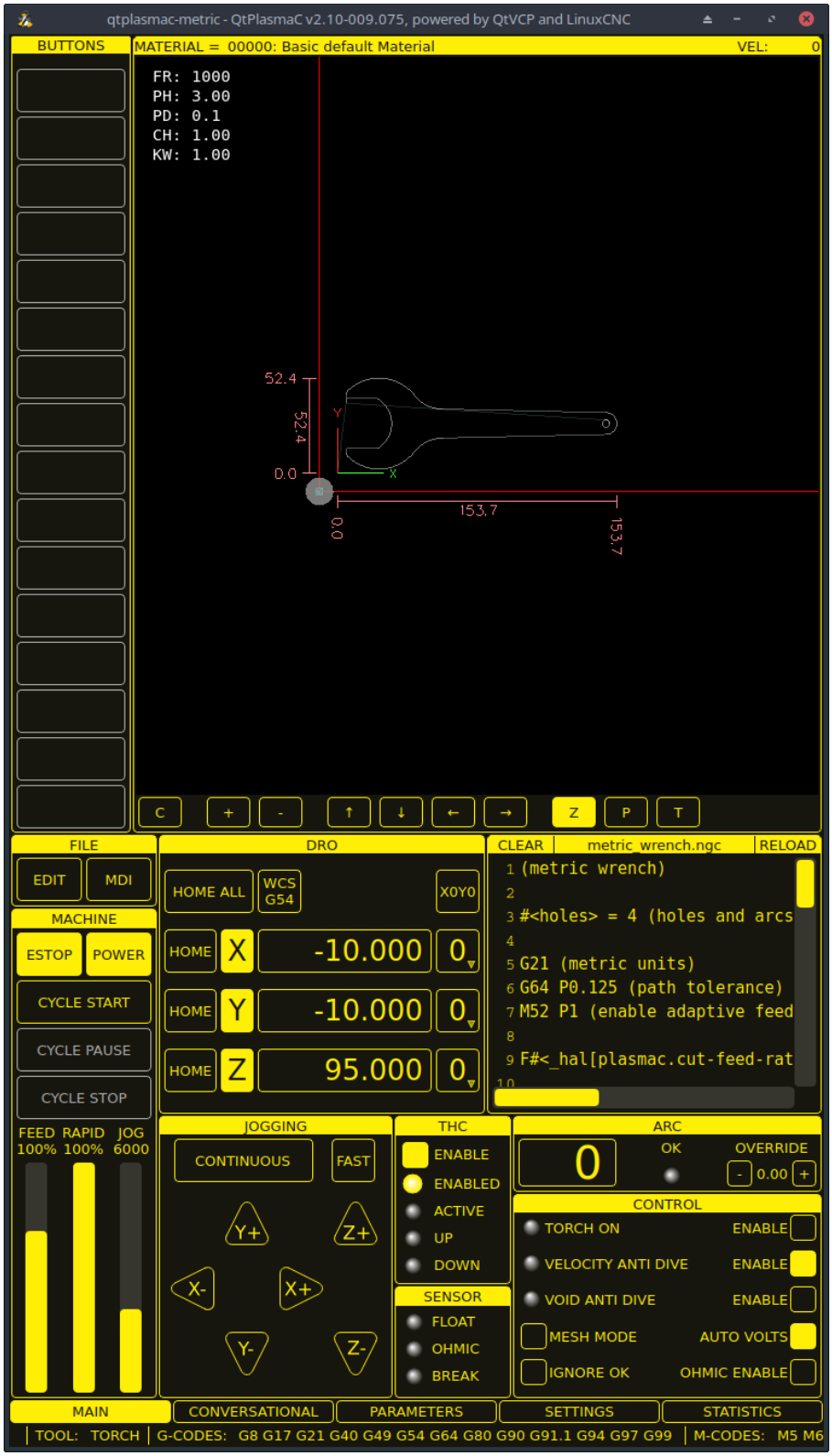


Figure 190. **9:16**

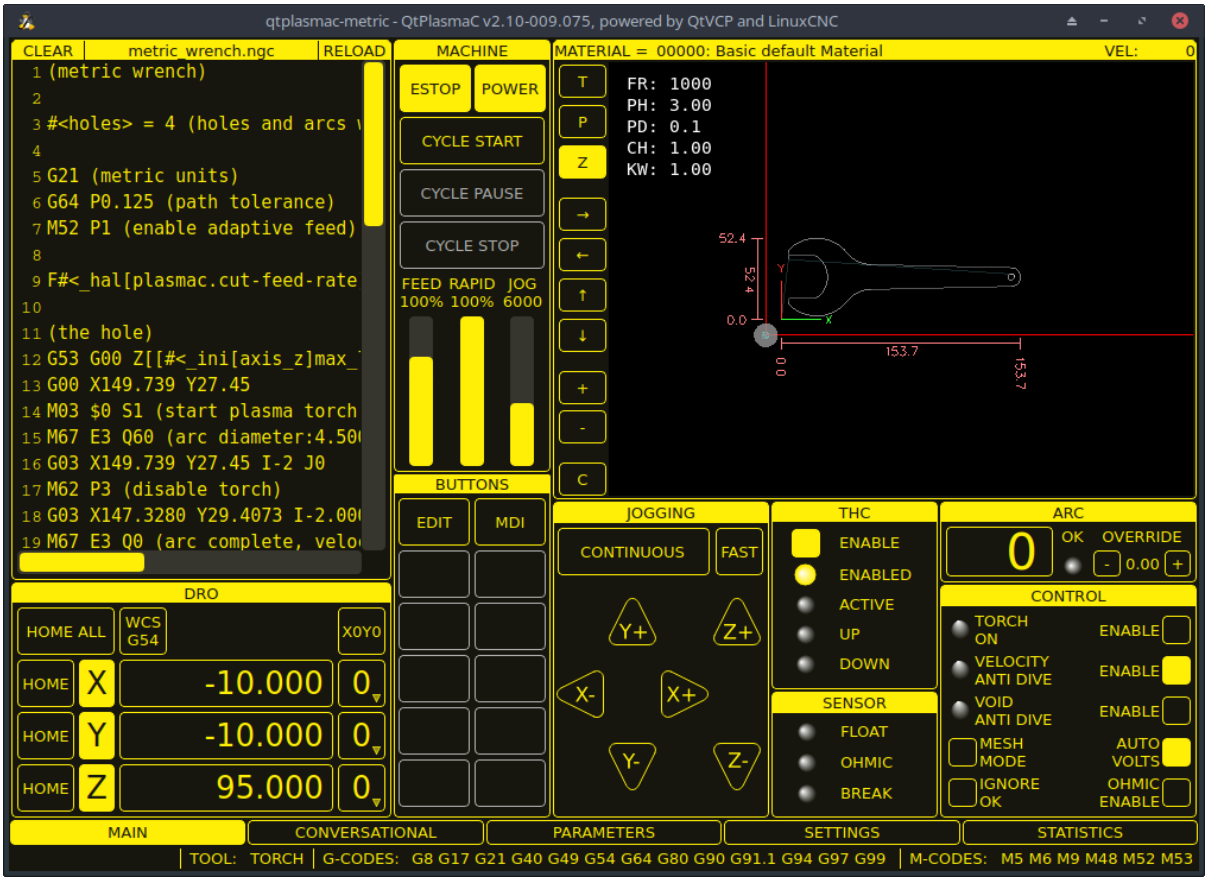


Figure 191. **4:3**

10.8.4. LinuxCNC installieren

Die bevorzugte Methode zur Installation von LinuxCNC ist über ein ISO-Image, wie unten beschrieben.

NOTE

Es ist möglich, LinuxCNC auf einer Vielzahl von Linux-Distributionen zu installieren und auszuführen; dies liegt jedoch außerhalb des Umfangs dieses Benutzerhandbuchs. Wenn der Benutzer eine andere Linux-Distribution als die empfohlenen installieren möchte, muss er zunächst seine bevorzugte Linux-Distribution installieren und anschließend die Master-Branch-Version von LinuxCNC (v2.10) sowie alle erforderlichen Abhängigkeiten einrichten. Es sollte außerdem beachtet werden, dass Bullseye die früheste von der Master-Branch-Version von LinuxCNC (v2.10) unterstützte Debian-Distribution ist. Buster wird nicht mehr unterstützt.

Wenn der Benutzer kein Linux installiert hat

Installationsanweisungen finden Sie hier.

Die Befolgung dieser Anweisungen wird eine Maschine mit dem aktuellen stabilen branch (Entwicklungs-Zweig) von LinuxCNC (v2.9) auf Debian 12 (Bookworm) ergeben. Der Benutzer muss dann den entsprechenden Anweisungen folgen, um zu der Version im master-branch von LinuxCNC (v2.10) zu aktualisieren.

Paket-Installation (Buildbot) wenn der Benutzer Linux mit Debian 12 (Bookworm) am Laufen hat

Folgen Sie den Anweisungen des Abschnitts "Update von LinuxCNC auf Debian Bookworm" in diesem Dokument.

Paket-Installation (Buildbot) Wenn der Benutzer Debian 12 (Bookworm) oder Debian 11 (Bullseye) nutzt

Eine Paketinstallation (Buildbot) verwendet vorgefertigte Pakete des LinuxCNC Buildbot.

Fügen Sie die GPG-Schlüssel hinzu und tragen Sie das Repository in der sources.list Datei ein, um die Debian-Version anzupassen.

Der nachfolgende Absatz würde den master branch (v2.10) für Bookworm hinzufügen.

```
deb     http://buildbot2.highlab.com/ bookworm master-uspace
```

Run In Place Installation, wenn der Benutzer bereits Linux installiert hat

Ein Run-in-Place-Installation läuft LinuxCNC aus einer lokal kompilierten Version in der Regel unter ~/linuxcnc-dev, Anweisungen für den Aufbau einer Run-in-Place-Installation sind hier verfügbar.

10.8.5. Erstellen einer QtPlasmaC Konfiguration

Vor der Erstellung einer QtPlasmaC-Konfiguration ist es wichtig, dass der Benutzer die verfügbaren Betriebsmodi sowie die für einen erfolgreichen Plasmabetrieb erforderlichen E/As genau kennt.

Modi

QtPlasmaC erfordert die Auswahl eines der folgenden drei Betriebsmodi:

Modus	Beschreibung
0	Verwendet einen externen Lichtbogenspannungseingang, um sowohl die Lichtbogenspannung (für die Brennerhöhensteuerung) als auch den Lichtbogen-OK zu berechnen.
1	Verwendet einen externen Lichtbogenspannungseingang zur Berechnung der Lichtbogenspannung (für die Brennerhöhensteuerung). Verwendet einen externen Lichtbogen-OK-Eingang für Lichtbogen-OK.
2	Verwendet einen externen Arc OK-Eingang für Arc OK. Verwendet externe Auf-/Ab-Signale für die Brennerhöhensteuerung.

IMPORTANT Wenn die Plasmastromquelle über einen Arc OK (Transfer)-Ausgang verfügt, wird empfohlen, diesen für Arc OK anstelle des weichen (berechneten) Arc OK zu verwenden, der von Modus 0 bereitgestellt wird. Es kann auch möglich sein, ein

Reed-Relais als alternative Methode zu verwenden, um ein Arc OK-Signal herzustellen, wenn die Stromquelle keines liefert.

NOTE Für die Feinabstimmung von Mode 0 Ark OK siehe Tuning Mode 0 Arc OK im Abschnitt Erweiterte Themen des Handbuchs.

Verfügbare I/Os

NOTE Dieser Abschnitt befasst sich nur mit den für QtPlasmaC erforderlichen Hardware-E/As. Die Anforderungen an die Basismaschine, wie Endschalter, Home-Schalter usw., kommen noch hinzu.

Name	Modi	Beschreibung
Lichtbogen-Spannung	0, 1	Analogeingang; **optional.** HAL-Pinname `plasmac.arc-voltage-in` Angeschlossen an den Geschwindigkeitsausgang einer mit Encoder ausgestatteten Breakout-Karte. Dieses Signal wird verwendet, um die Lichtbogenspannung abzulesen, um die notwendigen Korrekturen zu bestimmen, um den Brennerabstand zum Werkstück während des Schneidens einzuhalten.
Bogen OK	1, 2	Digitaler Eingang; **optional.** HAL-Pinname `plasmac.arc-ok-in` Angeschlossen vom Arc OK-Ausgang der Plasmastromquelle an einen Eingang auf der Breakout-Platine. Dieses Signal wird verwendet, um festzustellen, ob der Schneidlichtbogen hergestellt wurde und es für die Maschine in Ordnung ist, sich zu bewegen (manchmal auch Lichtbogentransfer genannt).
Schwimmerschalter (engl. float switch)	0, 1, 2	Digitaler Eingang; **optional, siehe Info-Tabelle unten:** HAL-Pin-Name `plasmac.float-switch` Wird von einem Breakout-Board-Eingang an einen Schalter am Schwebekopf angeschlossen. Dieses Signal wird verwendet, um das Werkstück mit dem Brenner mechanisch anzutasten und den Z-Nullpunkt an der Oberseite des Werkstücks zu setzen. Falls verwendet und kein ohmscher Taster konfiguriert ist, ist dies die Antastmethode. Wenn verwendet und ein ohmscher Messtaster konfiguriert ist, ist dies die Ausweichmethode.

Name	Modi	Beschreibung
Ohmsche Sonde (engl. ohmig probe)	0, 1, 2	Digitaler Eingang; **optional, siehe Info-Tabelle unten:** HAL-Pin-Name `plasmac.ohmic-probe` Wird vom Ausgang der ohmschen Sonde an den Eingang einer Breakout-Platine angeschlossen. Dieses Signal wird verwendet, um elektronisch zu tasten, indem ein Stromkreis mit dem Werkstück und den Brennerverbrauchsmaterialien geschlossen wird und Z-Null an der Oberseite des Werkstücks eingestellt wird. Falls verwendet, ist dies die primäre Antastmethode. Wenn eine ohmsche Sonde das Werkstück nicht findet und kein Schwimmerschalter vorhanden ist, wird die Abtastung fortgesetzt, bis der Brenner abreißt oder der minimale Z-Grenzwert erreicht wird.
Ohmsche Sonde aktivieren (engl. enable)	0, 1, 2	Digitaler Ausgang; **optional, siehe Info-Tabelle unten:** HAL-Pin-Name `plasmac.ohmic-enable` Wird von einem Breakout-Board-Ausgang mit einem Eingang verbunden, um die Leistung der ohmschen Sonde zu steuern.
Abreißschalter (engl. Breakaway Switch)	0, 1, 2	Digitaler Eingang; **optional, siehe Info-Tabelle unten:** + HAL Pin-Bezeichnung `plasmac.breakaway` Wird von einem Breakout-Board-Eingang an einen Schalter zur Erkennung des Abreißens eines Brenners angeschlossen. Dieses Signal erkennt, ob sich der Brenner von seiner Halterung (engl. cradle) gelöst hat.
Brenner ein (engl. torch on)	0, 1, 2	Digitaler Ausgang; **erforderlich.** HAL Pin-Bezeichnung `plasmac.torch-on` Wird von einem Breakout-Board-Ausgang an den "Brenner-Ein" (engl. torch on)-Eingang der Plasmastromversorgung angeschlossen. Dieses Signal wird zur Steuerung der Plasmastromversorgung und zum Zünden des Lichtbogens verwendet.
Nach oben bewegen	2	Digitaler Eingang; **optional.** HAL Pin-Bezeichnung `plasmac.move-up` Wird vom Aufwärtsausgang der externen THC-Steuerung an einen Eingang der Breakout-Karte angeschlossen. Dieses Signal wird verwendet, um die Z-Achse in einer Aufwärtsbewegung zu steuern und die notwendigen Korrekturen vorzunehmen, um den Brennerabstand zum Werkstück während des Schneidens beizubehalten.
Nach unten bewegen	2	Digitaler Eingang; **optional.** HAL Pin-Bezeichnung `plasmac.move-down` Wird vom Abwärtsausgang der externen THC-Steuerung an einen Breakout-Board-Eingang angeschlossen. Dieses Signal wird verwendet, um die Z-Achse in einer Abwärtsbewegung zu steuern und die notwendigen Korrekturen vorzunehmen, um den Brennerabstand zum Werkstück während des Schneidens beizubehalten.

Name	Modi	Beschreibung
Gravur aktivieren (engl. scribe arming)	0, 1, 2	Digitaler Ausgang; **optional.** HAL-Pin-Bezeichnung `plasmac.scribe-arm` Wird von einem Breakout-Board-Ausgang an die Schaltung für die Ritzschärfung angeschlossen. Dieses Signal wird verwendet, um den Ritzer auf dem Werkstück zu positionieren.
Gravieren starten (engl. Scribe On)	0, 1, 2	Digitaler Ausgang; **optional.** HAL-Pin-Bezeichnung `plasmac.scribe-on` Wird von einem Breakout-Board-Ausgang an die Scribe-On-Schaltung angeschlossen. Dieses Signal wird zum Einschalten des Ritzgeräts verwendet.
Laser an	0, 1, 2	Digitaler Ausgang; **optional.** HAL-Pin-Name `qtplasmac.laser_on` Dieses Signal wird verwendet, um den Ausrichtlaser einzuschalten.

Es ist nur einer der beiden Schalter **Float Switch** oder **Ohmic Probe** erforderlich. Wenn beide verwendet werden, dient **Float Switch** als Ausweichlösung, wenn **Ohmic Probe** nicht erkannt wird.

Wenn **Ohmic Probe** verwendet wird, muss **Ohmic Probe Enable** in der QtPlasmaC GUI aktiviert sein.

Der **Breakaway-Schalter** ist nicht zwingend erforderlich, da der **Float-Schalter** wie ein Breakaway-Schalter behandelt wird, wenn er nicht getestet wird. Wenn es sich um zwei separate Schalter handelt und nicht genügend Eingänge auf dem Breakout-Board vorhanden sind, können sie kombiniert und als **Float Switch** angeschlossen werden.

NOTE Die minimalen E/A-Anforderungen für eine funktionierende QtPlasmaC-Konfiguration sind: **Arc Voltage** Eingang ODER **Arc OK** Eingang, **Float Switch** Eingang und **Torch On** Ausgang. Um es noch einmal zu wiederholen: In diesem Fall behandelt QtPlasmaC den Schwimmerschalter als Abreißschalter, wenn er nicht sondiert wird.

Empfohlene Einstellungen:

Eine visuelle Darstellung der folgenden Begriffe finden Sie im Höhen-Diagramm.

- **[AXIS_Z] MIN_LIMIT** sollte knapp unter der Oberkante der Lamellen liegen, wobei der Weg des Schwimmerschalters und die Überfahrttoleranz berücksichtigt werden müssen. Wenn der Schwimmerschalter des Benutzers beispielsweise 4 mm braucht, um aktiviert zu werden, dann setzen Sie das Z-Minimum auf 5 mm (0,2") plus eine Toleranz für die Überschreitung (entweder nach der unten stehenden Gleichung berechnet oder 5 mm unterhalb der untersten Lamelle).
- **[AXIS_Z] MAX_LIMIT** sollte der höchste Wert sein, den der Benutzer für die Z-Achse verfahren soll (sie darf nicht niedriger sein als Z HOME_OFFSET).
- **[AXIS_Z] HOME** sollte auf einen Wert von ca. 5mm-10mm (0.2"-0.4") unter dem maximalen Grenzwert eingestellt werden.
- **Schwimmender Kopf** (engl. floating head) - es wird empfohlen, einen schwimmenden Kopf zu

verwenden, der ausreichend beweglich ist, um einen Überlauf während der Sondierung zu ermöglichen. Der Überlauf kann anhand der folgenden Formel berechnet werden:

```
o = 0.5 * a * (v / a)^2
```

wobei: o = Nachlauf (engl. overrun), a = Beschleunigung in Einheiten/s^2 und v = Geschwindigkeit in Einheiten/s.

Metrisches Beispiel: Bei einer MAX_ACCELERATION der Z-Achse von 600 mm/s^2 und einer MAX_VELOCITY von 60 mm/s würde der Nachlauf 3 mm betragen.

Kaiserliches Beispiel: Bei einer MAX_ACCELERATION der Z-Achse von 24 in/s^2 und einer MAX_VELOCITY von 2,4 in/s würde der Überlauf 0,12 in betragen.

Bei Maschinen, die einen ohmschen Messfühler als primäre Messmethode verwenden, wird dringend empfohlen, einen Schalter am Schwebekopf zu installieren, um die Z-Bewegung zu stoppen, falls der ohmsche Messfühler aufgrund von verschmutzten Oberflächen ausfällt.

Konfigurieren

LinuxCNC bietet zwei Konfigurationsassistenten, die zum Erstellen einer Maschinenkonfiguration verwendet werden können. Die Auswahl dieser Assistenten hängt von der Hardware ab, die zur Steuerung der Maschine verwendet wird.

Wenn der Benutzer eine Run-In-Place-Installation verwenden möchte, muss er vor dem Ausführen eines der folgenden Befehle den folgenden Befehl in einem Terminal ausführen:

```
source ~/linuxcnc-dev/scripts/rip-environment
```

Wenn Sie eine Paketinstallation verwenden, sind keine weiteren Maßnahmen erforderlich.

Wenn Sie eine parallele Schnittstelle verwenden, benutzen Sie den StepConf wizard, indem Sie den Befehl `stepconf` in einem Terminal-Fenster ausführen oder ihn über den Desktop-Menüeintrag **Anwendung → CNC → StepConf Wizard** starten.

Wenn Sie ein Mesa Electronics-Board verwenden, benutzen Sie den PnCconf wizard, indem Sie den Befehl `pncconf` in einem Terminal-Fenster ausführen oder ihn über den Desktop-Menüeintrag **Anwendung → CNC → PnCConf Wizard** starten.

Wenn Sie ein Pico Systems-Board verwenden, könnte Dieser LinuxCNC-Forumsthread hilfreich sein.

Die gerätespezifischen Einstellungen werden hier nicht beschrieben, sondern sind in der Dokumentation des jeweiligen Konfigurationsassistenten nachzulesen, der verwendet wird.

Für diese Assistenten gibt es LinuxCNC-Forumsbereiche:

StepConf Wizard

PnCconf Wizard

Füllen Sie die erforderlichen Einträge entsprechend der Konfiguration der Maschinenverdrahtung/Breakout-Platine aus.

QtPlasmaC fügt den LinuxCNC-Konfigurationsassistenten zwei Seiten für QtPlasmaC-spezifische Parameter hinzu, die beiden Seiten sind QtPlasmaC-Optionen und User Buttons. Füllen Sie jede der Assistenten QtPlasmaC Seite, um die Maschine, die konfiguriert wird und die Benutzer-Button-Anforderungen anzupassen.

Beachten Sie, dass die PnCConf-Optionen die Auswahl von Vorschub-Override, Lineargeschwindigkeit und Jog-Inkrementen durch den Benutzer erlauben, während diese in StepConf automatisch berechnet und eingestellt werden.

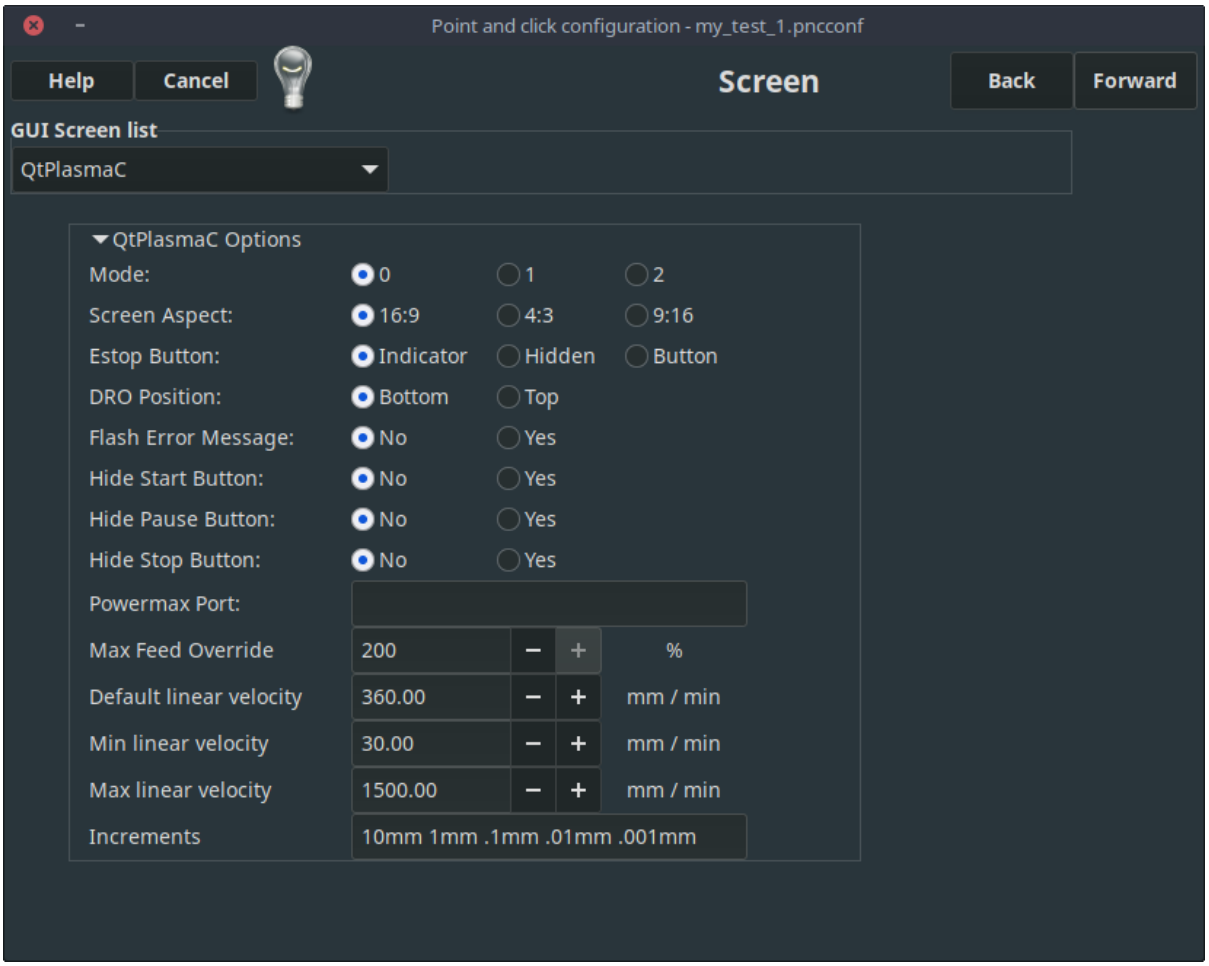


Figure 192. PnCconf QtPlasmaC Optionen

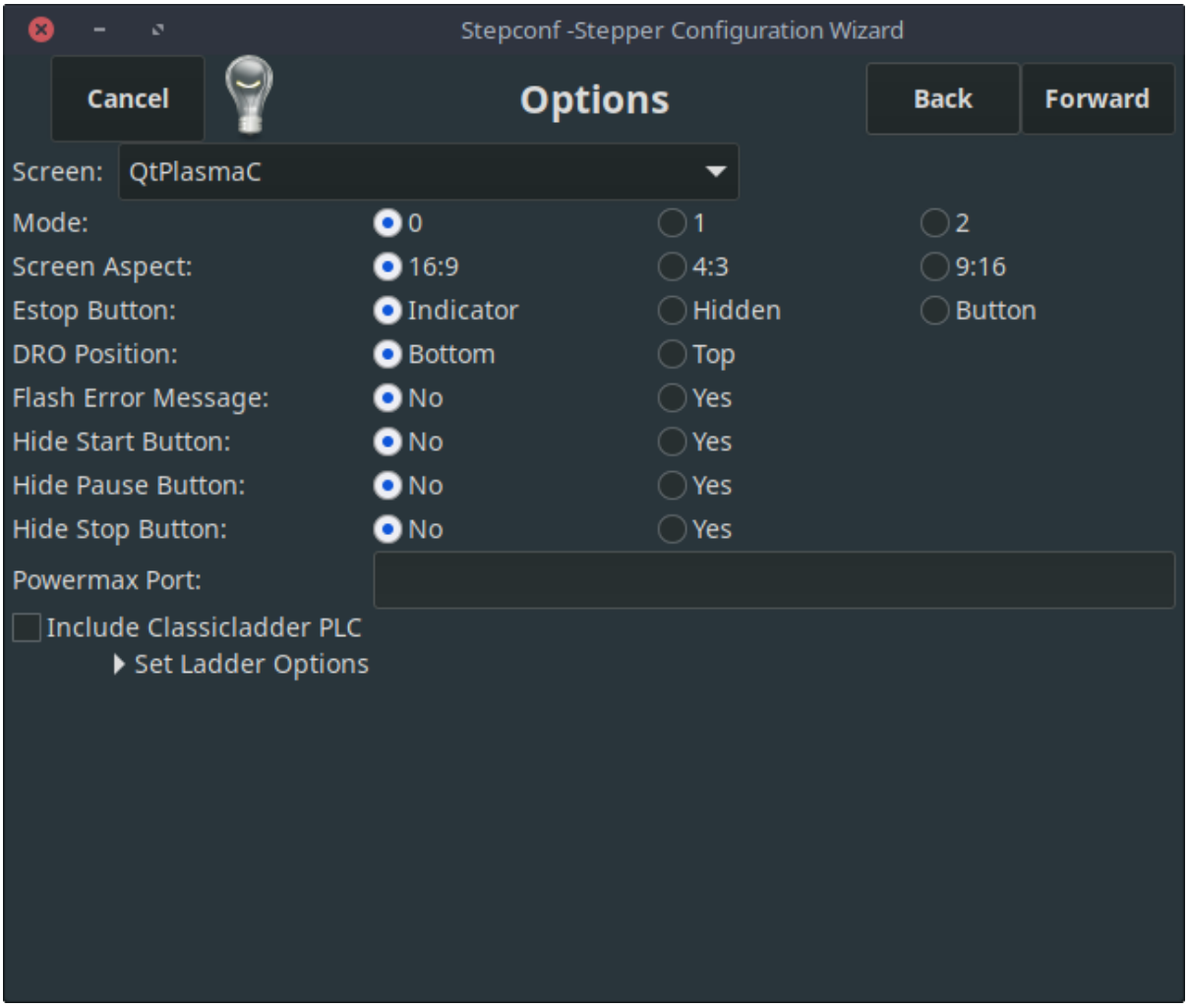


Figure 193. StepConf QtPlasmaC-Optionen

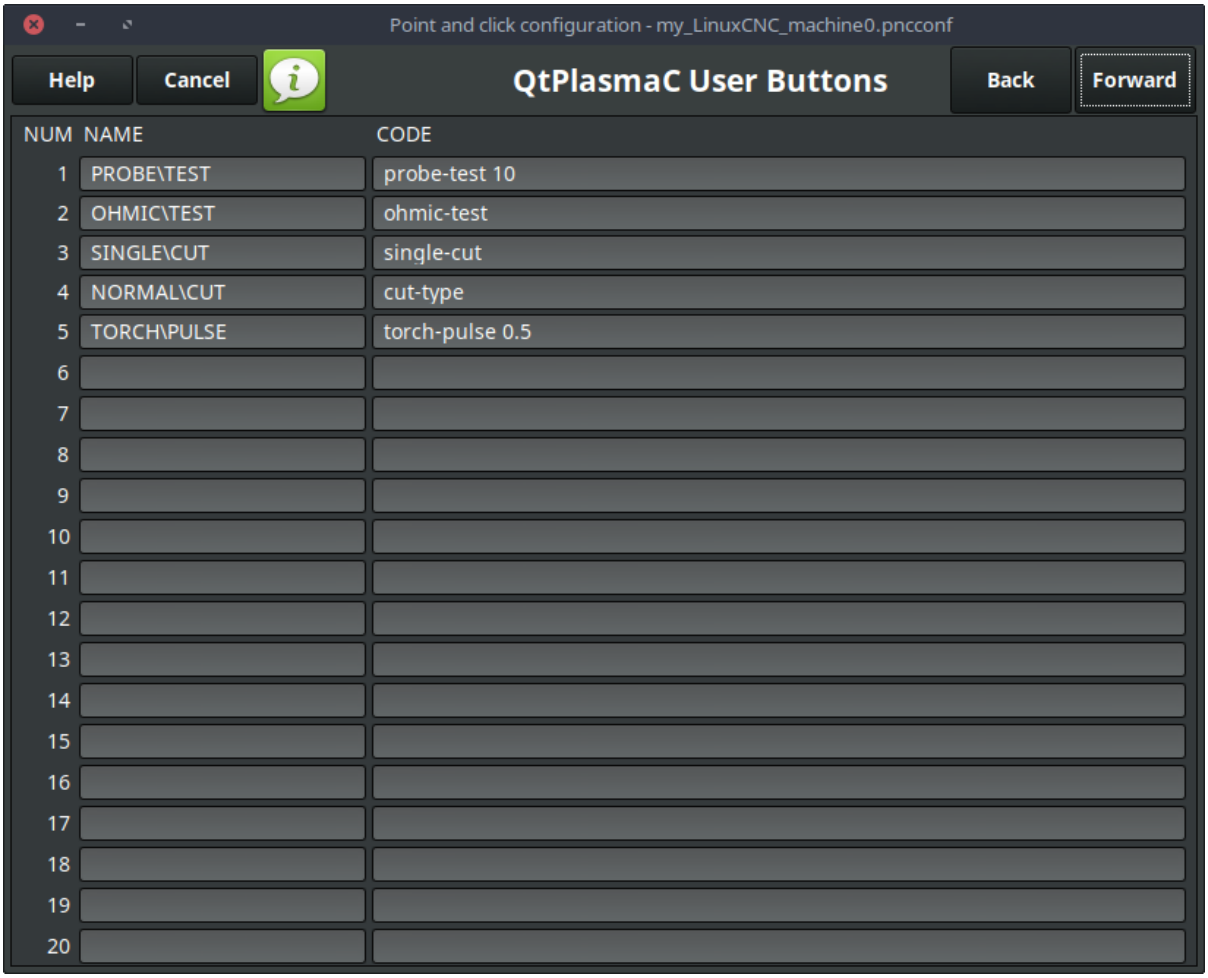


Figure 194. QtPlasmaC Benutzer-Buttons

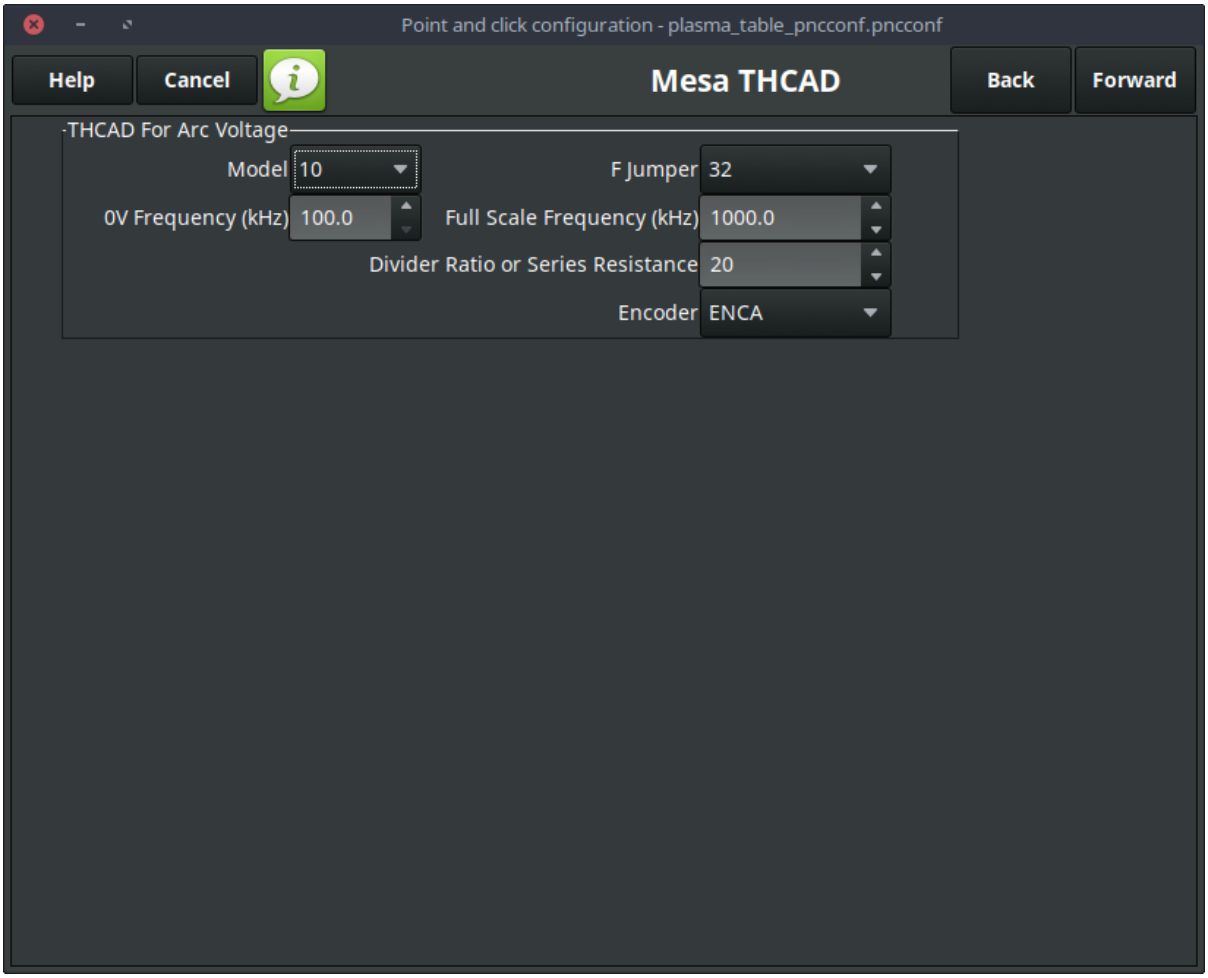


Figure 195. QtPlasmaC THCAD

Der THCAD-Bildschirm wird nur angezeigt, wenn im Kartenbildschirm ein Plasma-Encoder ausgewählt ist. Weitere Informationen finden Sie im Abschnitt dedicated über Mesa THCAD.

Wenn die Konfiguration abgeschlossen ist, speichert der Assistent eine Kopie der Konfiguration, die zu einem späteren Zeitpunkt geladen und bearbeitet werden kann. Eine funktionierende QtPlasmaC-Konfiguration wird im folgenden Verzeichnis erstellt: ~/linuxcnc/configs/<*Maschinenname*>.

Die Art und Weise, wie die neu erstellte QtPlasmaC-Konfiguration von der Kommandozeile aus ausgeführt werden kann, muss leicht der Installation von LinuxCNC angepasst werden:

Für eine Paketinstallation (Buildbot):

```
linuxcnc ~/linuxcnc/configs/_<machine_name>_/_<machine_name>_.ini
```

Für eine Installation an Ort und Stelle (engl. run-in-place):

```
~/linuxcnc-dev/scripts/linuxcnc ~/linuxcnc/configs/_<machine_name>_/_<machine_name>_.ini
```

Nach dem Ausführen des obigen Befehls sollte LinuxCNC mit der QtPlasmaC GUI sichtbar sein.

IMPORTANT	BEVOR DER BENUTZER FORTFÄHRT, SOLLTE ER IN DER LAGE SEIN, DIE MASCHINE IN DIE AUSGANGSPOSITION ZU BRINGEN, JEDE ACHSE AUF NULL ZU

> STELLEN, ALLE ACHSEN BIS ZU DEN WEICHEN GRENZWERTEN ZU VERFAHREN, OHNE DASS ES ZU EINEM ABSTURZ KOMMT, UND G-CODE-TESTPROGRAMME OHNE FEHLER AUSZUFÜHREN.

NUR WENN diese Kriterien erfüllt sind, sollte der Benutzer mit der Ersteinrichtung von QtPlasmaC fortfahren.

NOTE Es ist möglich, eine Simulationskonfiguration mit StepConf zu erstellen, aber es ist nicht möglich, Tandemgelenke in der Simulationskonfiguration zu haben.

Qt-Abhängigkeitsfehler

Wenn beim Versuch, die QtPlasmaC-Konfiguration auszuführen, Fehler in Bezug auf Qt-Abhängigkeiten auftreten, muss der Benutzer möglicherweise das QtVCP-Installationsskript ausführen, um diese Probleme zu beheben.

Geben Sie für eine Paketinstallation (Buildbot) den folgenden Befehl in einem Terminalfenster ein:

```
/usr/lib/python3/dist-packages/qtvcp/designer/install_script
```

Geben Sie für eine "run in place"-Installation den folgenden Befehl in ein Terminalfenster ein:

```
~/linuxcnc-dev/lib/python/qtvcp/designer/install_script
```

Erstmalige Einrichtung

The following heights diagrams will help the user visualize the different heights involved in plasma cutting and how they are measured. There are two different scenarios based on if the user chooses to use **Probe Height** only, or if the user chooses to use **Slat Height** AND **Material Thickness**.

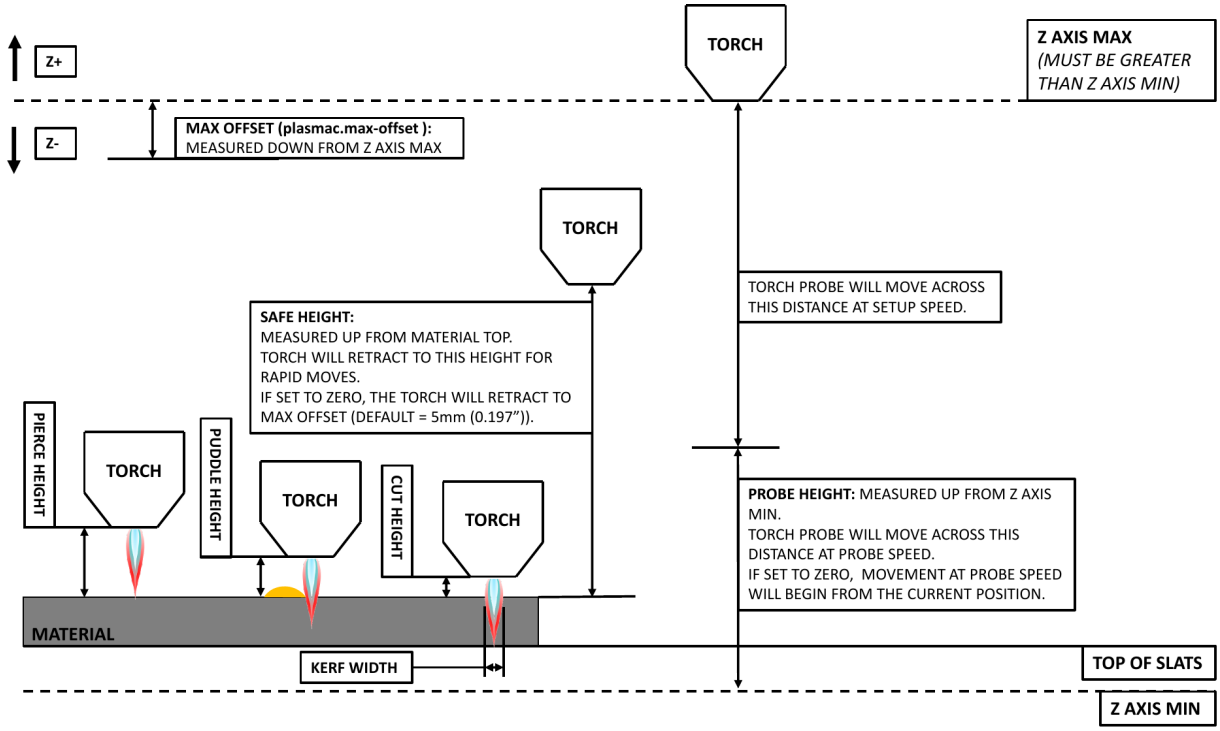


Figure 196. **Sondenhöhe allein**

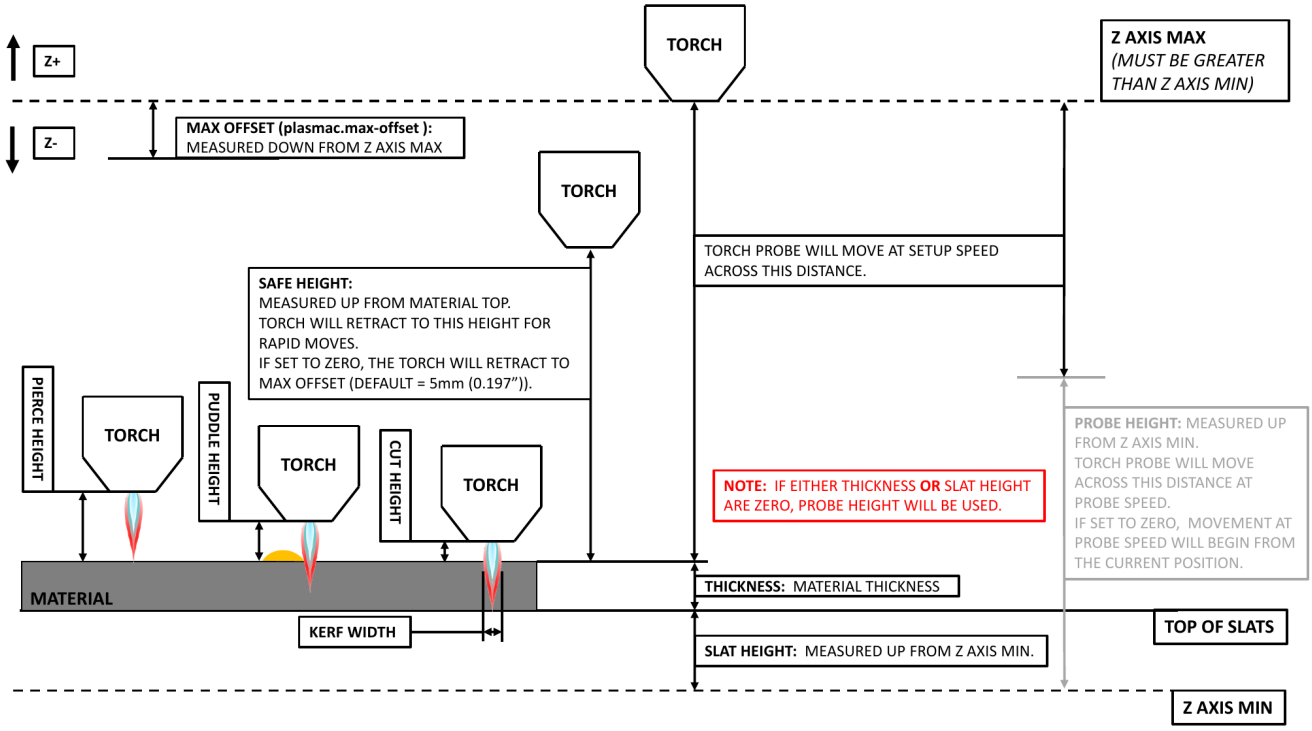


Figure 197. **Slat Height and Material Thickness**

Klicken Sie auf die Registerkarte Parameter um den Abschnitt **CONFIGURATION** anzuzeigen, in dem die vom Benutzer einstellbaren Parameter angezeigt werden. Es muss sichergestellt werden, dass jede dieser Einstellungen auf die Maschine zugeschnitten ist.

Um die Z-Achsen-DRO relativ zur Z-Achse MINIMUM_LIMIT einzustellen, sollte der Benutzer die folgenden Schritte durchführen. Es ist wichtig zu verstehen, dass in QtPlasmaC die Berührung der Z-Achsen-DRO keinen Einfluss auf die Z-Achsen-Position hat, während ein G-Code-Programm läuft. Diese

Schritte ermöglichen dem Benutzer lediglich eine einfachere Einstellung der Sondenhöhe, da nach Durchführung der Schritte der angezeigte Z-Achsen-DRO-Wert relativ zur Z-Achse MINIMUM_LIMIT ist.

NOTE Der Benutzer sollte mit den empfohlenen Z-Achsen Einstellungen vertraut sein.

1. Referenzierung der Z-Achse.
2. Vergewissern Sie sich, dass sich nichts unter dem Brenner befindet, dann bewegen Sie die Z-Achse nach unten, bis sie am MINIMUM_LIMIT der Z-Achse anhält, und klicken Sie dann auf die 0 neben der Z-Achsen-Anzeige, um die Z-Achse mit der ausgewählten Z-Achse auf Nullpunktverschiebung zu setzen. Dieser Schritt dient nur dazu, dem Benutzer eine einfachere Visualisierung und Einstellung der **Sondenhöhe** zu ermöglichen - dieser Wert wird vom MINIMUM_LIMIT der Z-Achse aufwärts gemessen.
3. Erneute Referenzierfahrt der Z-Achse.

Sonden-Test

Wenn das Gerät mit einem Schwimmerschalter ausgestattet ist, muss der Benutzer den Offset im Abschnitt **KONFIGURATION** auf der Registerkarte **PARAMETER** einstellen. Dies geschieht durch Ausführen eines "Probe Test"-Zyklus.

1. Überprüfen Sie, ob die Sondengeschwindigkeit und die Sondenhöhe im Abschnitt **CONFIGURATION** auf der Registerkarte **PARAMETERS** korrekt sind. QtPlasmaC kann mit der vollen Geschwindigkeit der Z-Achse tasten, solange die Maschine genügend Bewegung im Schwimmerschalter hat, um einen eventuellen Überlauf aufzufangen. Wenn die Maschine dafür geeignet ist, kann der Benutzer die Sondenhöhe auf einen Wert in der Nähe des Z-Achsen-Minimums einstellen und die gesamte Abtastung mit voller Geschwindigkeit durchführen.
2. Wenn die Maschine noch keine Referenzierfahrt durchführt und nicht in der Ausgangsposition ist, dann führen Sie die Referenzierfahrt durch.
3. Legen Sie etwas Material auf die Latten unter den Brenner.
4. Drücken Sie die Taste **PROBE TEST**.
5. Die Z-Achse tastet nach unten, findet das Material und bewegt sich dann nach oben auf die angegebene **Stechhöhe**, die durch das aktuell ausgewählte Material festgelegt ist. Der Brenner wartet in dieser Position für die in der Datei <*Maschinenname*>.prefs eingestellte Zeit. Die Standard-Haltezeit für die Sondenprüfung beträgt 10 Sekunden, dieser Wert kann in der Datei <*Maschinenname*>.prefs geändert werden. Danach kehrt der Brenner auf die Ausgangshöhe zurück.
6. Messen Sie den Abstand zwischen dem Material und der Brennerspitze, während der Brenner auf **Stechhöhe** (engl. pierce height) wartet.
7. Ist die Messung größer als die **Stechhöhe** (engl. pierce height) des aktuell ausgewählten Materials, dann verringern Sie den "Schwimmweg" (engl. float travel) im Abschnitt **KONFIGURATION** der Registerkarte **PARAMETER** um die Differenz zwischen dem gemessenen Wert und dem angegebenen Wert. Wenn der Messwert kleiner ist als die **Stechhöhe** des aktuell ausgewählten Materials, dann erhöhen Sie den "Schwimmweg" im Abschnitt **KONFIGURATION** der Registerkarte **PARAMETER** um die Differenz zwischen dem angegebenen Wert und dem Messwert.
8. Nachdem die Einstellungen für den "Schwimmweg" vorgenommen wurden, wiederholen Sie den Vorgang ab Nr. 4 oben, bis der gemessene Abstand zwischen dem Material und der Brennerspitze

mit der **Stechhöhe** des aktuell ausgewählten Materials übereinstimmt.

9. Wenn der Tisch mit einem Laser oder einer Kamera für die Blechausrichtung, einem Ritzgerät (engl. scribe) oder einer Versatz-Messung (engl. offset probing) ausgestattet ist, müssen die erforderlichen Offsets nach dem unter << peripheral-offsets,Peripherie-Offsets>> beschriebenen Verfahren angewendet werden. Die LASER- und/oder CAMERA-Schaltflächen werden erst angezeigt, wenn die entsprechenden Offset(s) vom Benutzer gesetzt und in der Datei <*Maschinenname*>.prefs gespeichert wurden.
10. HERZLICHEN GLÜCKWUNSCH! Der Benutzer sollte nun eine funktionierende QtPlasmaC Konfiguration haben.

NOTE Wenn die Zeitspanne zwischen dem Kontakt des Brenners mit dem Material und dem Zeitpunkt, zu dem der Brenner sich nach oben bewegt und auf der Durchstechhöhe zur Ruhe kommt, zu lang erscheint, finden Sie im << plasma:probing,Abschnitt Antasten>> eine mögliche Lösung.

IMPORTANT WENN EIN **Mesa Electronics THCAD** VERWENDET WIRD, DANN WURDE DER **Spannungsskalenwert** MATHEMATISCH ERREICHT. WENN DER BENUTZER beabsichtigt, SCHNITTSPANNUNGEN AUS EINEM SCHNITTSPLAN EINES HERSTELLERS ZU VERWENDEN, WÄRE ES RATSAM, MESSUNGEN DER TATSÄCHLICHEN SPANNUNGEN VORZUNEHMEN UND DIE **Spannungsskala** UND den **Spannungsoffset** FEINABSTIMMEN.

WARNING PLASMASCHNEIDSPANNUNGEN KÖNNEN TÖDLICH SEIN; WENN DER BENUTZER KEINE ERFAHRUNG MIT DIESEN MESSUNGEN HAT, SOLLTE ER SICH QUALIFIZIERTE HILFE HOLEN.

10.8.6. Umstellung auf QtPlasmaC von PlasmaC (AXIS oder GMOCCAPY)

Die automatisierte Migration auf QtPlasmaC von PlasmaC wird nicht mehr unterstützt. Der Benutzer muss entweder die PlasmaC-Konfiguration manuell umwandeln oder eine neue Konfiguration mit der Konfigurations-Assistenz (engl. configuration wizard) erstellen.

10.8.7. Andere Überlegungen zur Einrichtung von QtPlasmaC

Tiefpass-Filter

Die plasmac HAL-Komponente verfügt über einen eingebauten Tiefpassfilter, der bei Verwendung auf den **plasmac.arc-voltage-in**-Eingangspin angewendet wird, um jegliches Rauschen zu filtern, das fehlerhafte Spannungsmesswerte verursachen könnte. Der Tiefpassfilter sollte nur verwendet werden, nachdem mit Halscope die erforderliche Frequenz bestimmt wurde und festgestellt wurde, ob die Amplitude des Rauschens groß genug ist, um irgendwelche Probleme zu verursachen. Für die meisten Plasmageräte ist der Tiefpass nicht erforderlich und sollte nur verwendet werden, wenn er erforderlich ist.

Der diesem Filter zugewiesene HAL-Pin heißt **plasmac.lowpass-frequency** und ist standardmäßig auf 0

(deaktiviert) eingestellt. Um einen Tiefpassfilter auf die Lichtbogenspannung anzuwenden, würde der Benutzer den folgenden Eintrag in der Datei custom.hal im Konfigurationsverzeichnis des Geräts bearbeiten, um die entsprechende Grenzfrequenz, gemessen in Hertz (Hz), hinzuzufügen.

Zum Beispiel:

```
setp plasmac.lowpass-frequency 100
```

Das obige Beispiel würde eine Grenzfrequenz von 100 Hz ergeben.

Kontaktprellen

Das Prellen von Kontakten durch mechanische Relais, Schalter oder externe Störungen kann ein uneinheitliches Verhalten der folgenden Schalter verursachen:

- Schwimmerschalter (engl. float switch)
- Ohmsche Sonde (engl. ohmig probe)
- Abreißschalter (engl. Breakaway Switch)
- Lichtbogen OK (für Modi 1 & 2)

Da die Software in der Lage ist, Abtastraten zu erzielen, die schneller sind als die Kontaktprellzeit, ist es möglich, dass die Software das Kontaktprellen als mehrere Änderungen der Eingangszustände innerhalb eines sehr kurzen Zeitraums ansieht und dies fälschlicherweise als ein sehr schnelles Ein-Aus des Eingangs interpretiert. Eine Methode zur Abschwächung des Kontaktprellens ist das "Entprellen" des Eingangs. Zusammenfassend lässt sich sagen, dass der Eingangszustand für aufeinanderfolgende Verzögerungsperioden auf dem entgegengesetzten Zustand des Ausgangszustands stabil sein muss, bevor sich der Zustand des Ausgangs ändert.

Die Entprellungszeit kann durch Bearbeiten des entsprechenden Entprellungswerts in der Datei custom.hal im <*Machinenname*> Konfigurationsverzeichnis geändert werden.

Jeder Schritt der Verzögerung fügt der Entprellzeit einen Servo-Thread-Zyklus hinzu. Beispiel: Bei einer Servo-Thread-Periode von 1000000 (gemessen in Nanosekunden) entspricht eine Entprellverzögerung von 5 somit 5000000 ns oder 5 ms.

Bei Schwimmer- und ohmschen Schaltern entspricht dies einer Erhöhung der Abtasthöhe um 0,001 mm (0,00004").

Es wird empfohlen, die Entprellungswerte so niedrig wie möglich zu halten und dennoch konsistente Ergebnisse zu erzielen. Die Verwendung von Halscope zur Aufzeichnung der Eingaben ist eine gute Möglichkeit, den richtigen Wert zu ermitteln.

Für QtPlasmaC-Installationen wird das Entprellen durch die Verwendung der HAL dbounce Konponente erreicht, der eine spätere Alternative zur ursprünglichen `debounce-Komponente darstellt. Diese neue Version ermöglicht das Laden und Benennen einzelner Entprell-Instanzen und ist mit der Verarbeitung von Twopass HAL-Dateien kompatibel.

Alle vier oben genannten Signale haben eine individuelle Entprellungskomponente, so dass die

Entprellungszeiten individuell auf jeden Eingang abgestimmt werden können. Spätere Updates von QtPlasmaC werdeb Änderungen, die an diesen Werten in der Datei custom.hal vorgenommen werden, nicht überschreiben.

Die Standardverzögerung für alle vier Eingänge beträgt fünf Servogewindeperioden. In den meisten Fällen wird dieser Wert gut funktionieren. Wenn einer der Eingänge keine mechanischen Schalter verwendet, kann es möglich sein, die Verzögerung für diese Eingänge entweder zu reduzieren oder zu entfernen.

Wenn die Entprellung für andere Geräte wie z. B. Heim- oder Endschalter usw. erforderlich ist, können weitere dbounce-Komponenten in einer der HAL-Dateien hinzugefügt werden, ohne die hier aufgeführten Signale zu berücksichtigen.

Kontakt Last

Mechanische Relais und Schalter benötigen in der Regel einen Mindeststrom, der durch die Kontakte fließt, um zuverlässig zu funktionieren. Dieser Strom variiert je nach dem Material, aus dem die Kontakte des Geräts hergestellt sind.

Je nach dem angegebenen Mindestkontaktstrom und dem vom Eingabegerät aufgenommenen Strom kann es erforderlich sein, eine Methode zur Erhöhung des Stroms durch die Kontakte vorzusehen.

Die meisten Relais mit Goldkontakten benötigen keinen zusätzlichen Strom für einen zuverlässigen Betrieb.

Es stehen zwei verschiedene Methoden zur Verfügung, um diesen Mindeststrom bereitzustellen, falls dies erforderlich ist:

1. Ein 0,1 µF Folienkondensator über den Kontakten.
2. Ein 1200 Ω 1 W-Widerstand über der Last (siehe nachfolgende Berechnungen).

Schaltpläne sind unter Kontakt Last Schaltplan (engl. contact load schematics) zu finden.

Weitere Informationen zur Kontaktschaltlast finden Sie auf Seite VI des Finder Allgemeine Technische Informationen Dokuments.

Berechnungen:

Bei Verwendung einer Mesa-Karte beträgt der Eingangswiderstand eines 7I96 4700 Ω (konsultieren Sie immer das zur verwendeten Revision gehörende Produkthandbuch, da diese Werte manchmal von Revision zu Revision variieren), was einen Kontaktstrom von 5,1 mA bei einer Versorgungsspannung von 24 V ergibt (I = U/R). [1].

Als Beispiel: Das typische Relais, das in einem Hypertherm Powermax 65 Plasmaschneider verwendet wird (TE T77S1D10-24), erfordert eine minimale Kontaktbelastung von 100 mA bei 5 VDC, was 0,5 W (P = I * V) abgibt. Bei Verwendung einer 24 VDC-Stromversorgung würde dies einem Mindeststrom von 20,8 mA entsprechen. Da der vom Mesa-Eingang aufgenommene Strom geringer ist als der vom Relais benötigte, muss der Strom erhöht werden.

Der Widerstand kann mit $R = U_s / (I_m - I_i)$ berechnet werden, wobei:

- R = berechneter Widerstand
- U*s* = Versorgungsspannung
- I_m = erforderlicher Mindeststrom
- I_i = Eingangsstrom

Die Verwendung eines 7I96 mit einem Eingangsstrom von 5,1 mA ergibt einen berechneten Wert von 1529 Ω (= 24 V / (.0208 - .0051) A). Dieser Wert könnte dann auf einen handelsüblichen 1500 Ω-Widerstand abgerundet werden, was eine kleine Sicherheitsspanne ergibt.

Die Verlustleistung kann mit $P = U_s^2 / R_s$ berechnet werden, wobei:

- P = Leistung
- U*s* = Versorgungsspannung
- R_s = gewählter Widerstand

Dies ergibt einen Wert von 0,38 W. Dieser Wert könnte dann auf 1 W aufgerundet werden, was eine gute Sicherheitsmarge ergibt. Die endgültige Wahl wäre ein 1500 Ω 1 W-Widerstand.

Desktop-Starthilfe

Wenn bei der Erstellung der Konfiguration kein Link zum Starten der Konfiguration erstellt wurde, kann der Benutzer einen Desktop-Launcher für die Konfiguration erstellen, indem er mit der rechten Maustaste auf den Desktop klickt und Launcher erstellen o.ä. wählt. Daraufhin wird ein Dialogfeld zum Erstellen eines Launchers angezeigt. Geben Sie dem Symbol einen schönen kurzen Namen, geben Sie etwas für den Befehl ein und klicken Sie auf OK.

Nachdem der Launcher auf dem Desktop erscheint, klicken Sie mit der rechten Maustaste darauf und bearbeiten Sie ihn mit dem Editor Ihrer Wahl. Bearbeiten Sie die Datei so, dass sie ungefähr so aussieht:

```
[Desktop Entry]
Comment=
Terminal=false
Name=LinuxCNC
Exec=sh -c "linuxcnc $HOME/linuxcnc/configs/<machine_name>/<machine_name>.ini"
Type=Application
Icon=/usr/share/pixmaps/linuxcncicon.png
```

Wenn der Benutzer ein Terminalfenster hinter dem GUI-Fenster öffnen möchte, ändern Sie die Terminal-Zeile in:

```
Terminal=true
```

Die Anzeige eines Terminals kann für Fehler- und Informationsmeldungen nützlich sein.

QtPlasmaC Dateien

Nach einer erfolgreichen QtPlasmaC-Installation werden die folgenden Dateien im

Konfigurationsverzeichnis angelegt:

Dateiname	Funktion
<*Maschinenname*>.ini	Konfigurationsdatei für den Rechner.
<*Maschinenname*>.hal	HAL für die Maschine.
<*Maschinenname*>.prefs	Konfigurationsdatei für QtPlasmaC-spezifische Parameter und Einstellungen.
custom.hal	HAL-Datei für Benutzeranpassungen.
custom_postgui.hal	HAL-Datei zur Benutzeranpassung, die ausgeführt wird, nachdem die GUI initialisiert wurde.
shutdown.hal	HAL-Datei, die während der Abschaltsequenz ausgeführt wird.
tool.tbl	Werkzeugtabelle zum Speichern von Offset-Informationen für zusätzliche Werkzeuge (gravieren usw.), die von der QtPlasmaC-Konfiguration verwendet werden.
qtplasmac	Link zu dem Verzeichnis, das die üblichen QtPlasmaC-Unterstützungsdateien enthält.
Backup	Verzeichnis für Sicherungskopien der Konfigurationsdateien.

NOTE <*Maschinenname*> ist der Name, den der Benutzer in das Feld "Maschinenname" des Konfigurationsassistenten eingegeben hat.

NOTE Benutzerdefinierte Befehle sind in den Dateien custom.hal und custom_postgui.hal zulässig, da sie bei Aktualisierungen nicht überschrieben werden.

Nach der ersten Ausführung einer neuen Konfiguration werden die folgenden Dateien im Konfigurationsverzeichnis erstellt:

Dateiname	Funktion
<machine_name>_material.cfg	Datei zum Speichern der Materialeinstellungen aus dem Abschnitt MATERIAL der PARAMETER Registerkarte.
update_log.txt	Datei zum Speichern von Protokoll der wichtigsten Updates. Wichtige (engl. major) Updates sind solche, die eine Änderung der Konfiguration eines Benutzers vornehmen.
qtvcp.prefs	Datei mit den QtVCP-Einstellungen.
qtplasmac.qss	Datei, die das Stylesheet für die aktuell geladene Sitzung von QtPlasmaC speichert.

NOTE Die Konfigurationsdateien (<*Maschinenname*>.ini und <*Maschinenname*>.hal), die vom Konfigurationsassistenten erstellt werden, erläutern die Anforderungen in Kommentare, auch um die manuelle Bearbeitung dieser Konfigurationen zu erleichtern.

Sie können mit einem beliebigen Texteditor bearbeitet werden.

NOTE Die Datei <*Maschinenname*>.prefs ist reiner Text und kann mit jedem Texteditor bearbeitet werden.

INI-Datei

QtPlasmaC hat einige spezifische Variablen in der Datei <*Maschinenname*>.ini wie folgt:

[FILTER] Abschnitt

Diese Variablen sind erforderlich.

```
PROGRAM_EXTENSION = .ngc,.nc,.tap G-code File (*.ngc, *.nc, *.tap)
ngc               = qtplasmac_gcode
nc                = qtplasmac_gcode
tap               = qtplasmac_gcode
```

[RS274NGC] Abschnitt

Diese Variablen sind erforderlich.

```
RS274NGC_STARTUP_CODE = G21 G40 G49 G80 G90 G92.1 G94 G97 M52P1
SUBROUTINE_PATH       = ./:../../nc_files
USER_M_PATH           = ./:../../nc_files
```

NOTE Für eine imperiale Konfiguration ersetzen Sie G21 oben durch G20.

NOTE Beide oben genannten Pfade zeigen die Mindestanforderungen.

IMPORTANT SIEHE PFADTOLERANZ FÜR RS274NGC_STARTUP_CODE INFORMATIONEN ZU G64.

[HAL] Abschnitt

Diese Variablen sind erforderlich.

```
HALUI           = halui (required)
HALFILE         = _<machine_name>_.hal (the machine HAL file)
HALFILE         = plasmac.tcl (the standard QtPlasmaC HAL file)
HALFILE         = custom.hal (Users custom HAL commands)
POSTGUI_HALFILE = postgui_call_list.hal (required)
SHUTDOWN        = shutdown.hal (shutdown HAL commands)
```

NOTE Der Benutzer kann benutzerdefinierte HAL-Befehle in der Datei custom.hal platzieren, da diese Datei nicht durch QtPlasmaC-Updates überschrieben wird.

[DISPLAY] Abschnitt

Diese Variable ist erforderlich.

```
DISPLAY = qtvcp qtplasmac      (use 16:9 resolution)
        = qtvcp qtplasmac_9x16 (use 9:16 resolution)
        = qtvcp qtplasmac_4x3  (use 4:3 resolution)
```

Es gibt mehrere QtVCP-Optionen, die hier beschrieben werden: QtVCP INI Einstellungen

Zum Beispiel würde das Folgende einen QtPlasmaC-Bildschirm mit 16:9-Auflösung im Vollbildmodus starten:

```
DISPLAY = qtvcp -f qtplasmac
```

[TRAJ] Abschnitt

Diese Variable ist erforderlich.

```
SPINDELN = 3
```

[AXIS_X] Abschnitt

Diese Variablen sind erforderlich.

```
MAX_VELOCITY = das Doppelte des Wertes in dem entsprechenden Gelenk
MAX_ACCELERATION = das Doppelte des Wertes in dem entsprechenden Gelenk
OFFSET_AV_RATIO = 0.5
```

[AXIS_Y] Abschnitt

Diese Variablen sind erforderlich.

```
MAX_VELOCITY = das Doppelte des Wertes in dem entsprechenden Gelenk
MAX_ACCELERATION = das Doppelte des Wertes in dem entsprechenden Gelenk
OFFSET_AV_RATIO = 0.5
```

[AXIS_Z] Abschnitt

Diese Variablen sind erforderlich.

```
MIN_LIMIT = knapp unterhalb der Oberkante der Tischlatten
MAX_VELOCITY = das Doppelte des Wertes des entsprechenden Gelenks
MAX_ACCELERATION = das Doppelte des Wertes des entsprechenden Gelenks
OFFSET_AV_RATIO = 0,5
```

NOTE

Mit der Ausnahme von engl. tube cutting mit gewinkelter A, B oder C Achse, verwendet QtPlasmaC das LinuxCNC-Feature "Externe Offsets" für alle Z-Achsen-Bewegungen und für das Bewegen der X- und/oder Y-Achse für einen Verschleißteilwechsel im Pausenzustand. Für weitere Informationen über diese Funktion lesen Sie bitte External

> Axis Offsets in der LinuxCNC Dokumentation.

10.8.8. QtPlasmaC GUI Überblick

Die folgenden Abschnitte geben einen allgemeinen Überblick über das Layout von QtPlasmaC.

Beenden von QtPlasmaC

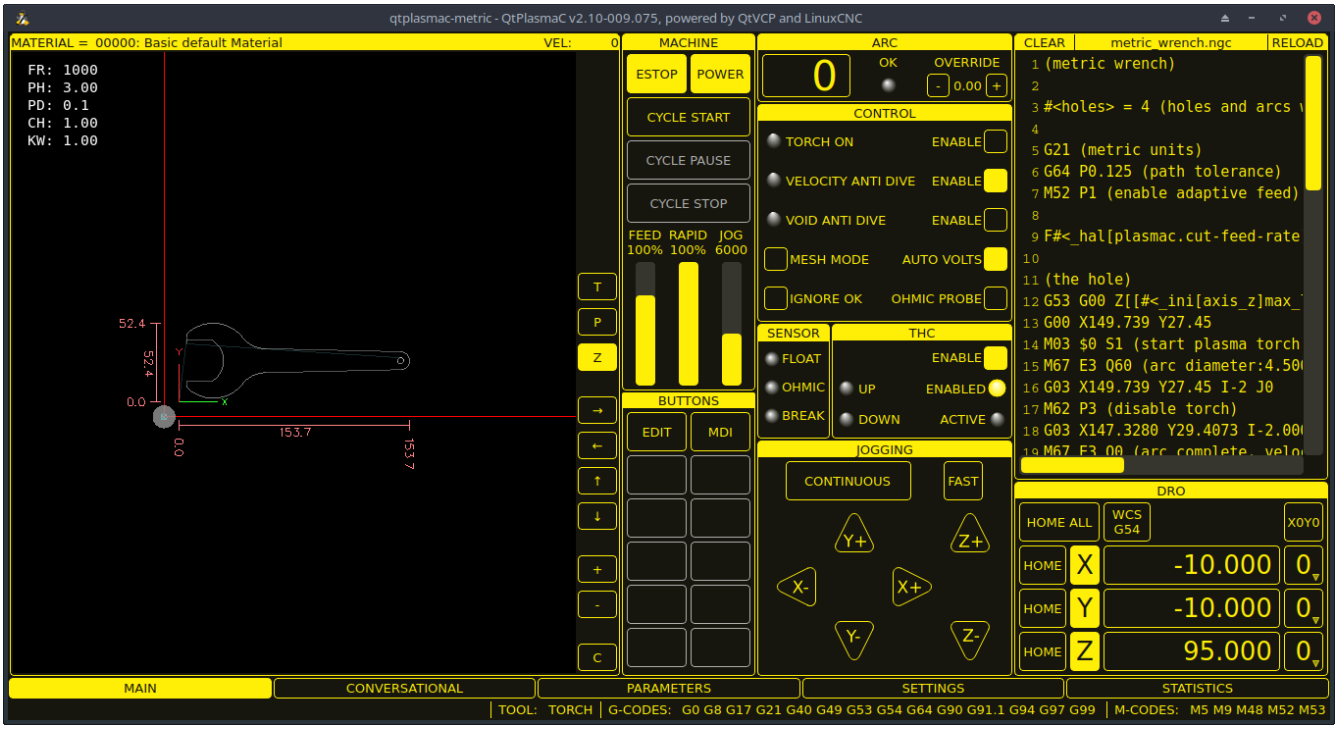
Das Beenden oder Herunterfahren von QtPlasmaC erfolgt entweder durch:

1. Klicken Sie auf die Schaltfläche zum Herunterfahren des Fensters in der Titelleiste des Fensters
2. Drücken Sie lange auf die Taste **POWER** auf der Haupt-Registerkarte (engl. main).

Eine Warnung beim Herunterfahren kann bei jedem Herunterfahren angezeigt werden, indem Sie das Kontrollkästchen **Beenden-Warnung** auf der Registerkarte mit EINSTELLUNGEN aktivieren.

HAUPT (engl. main)-Registerkarte (engl. tab)

Screenshot-Beispiel des QtPlasmaC Haupt-Registerkarte im Seitenverhältnis **16:9**:



Einige Funktionen/Merkmale werden nur für bestimmte Modi verwendet und werden nicht angezeigt, wenn sie für den gewählten QtPlasmaC-Modus nicht erforderlich sind.

Table 61. Attribute des **VORSCHAU-FENSTERS**

Name	Beschreibung
Material	In diesem Bereich kann die obere Kopfzeile angeklickt werden, um ein Dropdown-Menü zu öffnen. Es wird verwendet, um die aktuellen Materialschnittparameter manuell auszuwählen. Wenn in der Materialdatei keine Materialien vorhanden sind, wird nur das Standardmaterial angezeigt.
Geschw.:	Zeigt die aktuelle Vorschubgeschwindigkeit des Tisches an, einschließlich Eil- und Schnittbewegungen. Während des Schneidens aktualisiert sich diese Anzeige, falls eine Geschwindigkeits-Reduction aktiv ist, und gibt den Prozentsatz der ursprünglich programmierten Vorschubgeschwindigkeit an. Beispiel: „VEL@20%:“ bedeutet, dass der Tisch mit 20 % der programmierten Vorschubgeschwindigkeit schneidet – also einer Reduktion um 80 %.
FR:	Wenn "View Material" auf der PARAMETER-Registerkarte ausgewählt ist, wird die Vorschubgeschwindigkeit des aktuell ausgewählten Materials angezeigt.
PH:	Wenn auf der Registerkarte mit EINSTELLUNGEN die Option "View Material" ausgewählt ist, wird hier die Durchstechhöhe des aktuell ausgewählten Materials angezeigt.
PD:	Wenn auf der Registerkarte Einstellungen die Option "Material-Ansicht" ausgewählt ist, wird hier die Durchstoßverzögerung des aktuell ausgewählten Materials angezeigt.
CH:	Wenn auf der Registerkarte Einstellungen die Option "Material-Ansicht" ausgewählt ist, wird die Schnitthöhe des aktuell ausgewählten Materials angezeigt.
CA:	Wenn auf der Registerkarte EINSTELLUNGEN (engl. settings) die Option "View Material" (Anzeige von Material) ausgewählt ist und die RS485-Kommunikation aktiviert ist, wird hier die Stromstärke des aktuell ausgewählten Materials angezeigt.
T	Diese Schaltfläche ändert die preview in eine vollständige Tabellenansicht von oben nach unten.
P	Diese Schaltfläche ändert die preview in eine isometrische Ansicht.
Z	Diese Schaltfläche ändert die preview in eine Ansicht von oben nach unten.
→	Diese Schaltfläche verschiebt die Voransicht (engl. preview) nach rechts.
←	Diese Schaltfläche schwenkt die Voransicht nach links.
	Diese Schaltfläche schwenkt die Voransicht nach oben.
	Diese Schaltfläche schwenkt die Voransicht nach unten.
+	Diese Schaltfläche vergrößert die Voransicht.
-	Diese Schaltfläche vergrößert die Voransicht.
C	Diese Schaltfläche löscht die Live-Darstellung.

Table 62. **MASCHINEN** *Darstellung*

Name	Beschreibung
ESTOP (engl. für Notaus)	Wenn ESTOP_TYPE = 0 in der **[GUI_OPTIONS]** Sektion der Datei <*Maschinenname*>.prefs hinzugefügt wird, zeigt diese Schaltfläche nur den Status des Hardware Notaus (engl. E-stops) an. Wenn ESTOP_TYPE = 1 in der **[GUI_OPTIONS]** Sektion der Datei <*Maschinenname*>.prefs hinzugefügt wird, wird diese Schaltfläche nicht angezeigt. Wenn ESTOP_TYPE = 2 in der **[GUI_OPTIONS]** Sektion der Datei <*Maschinenname*>.prefs hinzugefügt wird, fungiert diese Schaltfläche als GUI E-stop.
POWER (engl. für Leistung oder Strom)	Diese Schaltfläche schaltet die GUI ein und erlaubt QtPlasmaC/LinuxCNC die Steuerung der Hardware. Wenn Sie die **POWER**-Taste länger als zwei Sekunden gedrückt halten, wird ein Dialog zum Beenden der QtPlasmaC-Anwendung angezeigt.
ZYKLUSSTART	Mit dieser Button startet den Zyklus für jede geladene G-Code-Datei.
ZYKLUSPAUSE	Mit diesem Button wird der Zyklus für jede geladene G-Code-Datei angehalten. Wenn ein Zyklus pausiert wird, zeigt dieser Button "ZYKLUS FORTSETZEN (engl. CYCLE RESUME) an und blinkt. Durch Drücken von **ZYKLUS FORTSETZEN** wird der Zyklus fortgesetzt.
ZYKLUS STOP (engl. cycle stop)	Mit dieser Taste wird ein aktiv laufender oder pausierter Zyklus gestoppt. Dies beinhaltet: - G-Code-Programme - Brennerimpuls, wenn der Impuls während **CYCLE PAUSE** gestartet wurde (dies bricht auch die pausierte G-Code-Programmausführung ab) - Sonden-Test - Einrahmung - Manueller Schnitt
FEED	Dieser Schieberegler übersteuert die Vorschubgeschwindigkeit für alle Vorschubbewegungen. Jeder andere Wert als 100% lässt das Label blinken. Ein Klick auf das Label stellt den Schieberegler wieder auf 100%.
RAPID	Dieser Schieberegler überschreibt die Schnellrate für alle Schnellbewegungen. Jeder andere Wert als 100 % führt dazu, dass das Label blinkt. Ein Klick auf das Label stellt den Schieberegler wieder auf 100%.
JOG	Mit diesem Schieberegler wird die Tippgeschwindigkeit eingestellt. Wenn Sie auf das Etikett klicken, wird der Schieberegler auf die lineare Standardgeschwindigkeit zurückgesetzt, wie sie in der Datei <*machine_name*>.ini eingestellt ist.

SCHALTFLÄCHEN *(engl. buttons)*

Die Panele enthalten Schaltflächen (Buttons) für die Bedienung der Maschine.

Die Schaltflächen **EDIT** und **MDI** sind permanent, alle anderen Schaltflächen sind vom Benutzer in der Datei <*machine_name*>.prefs programmierbar.

Siehe custom user buttons für detaillierte Informationen über benutzerdefinierte Schaltflächen.

Name	Beschreibung
EDIT	Diese Schaltfläche öffnet einen G-Code-Editor für das aktuell geladene Programm.
MDI	Diese Schaltfläche versetzt QtPlasmaC in den manuellen Dateneingabemodus (MDI), der die MDI HISTORY und ein Eingabefeld über dem G-Code-Fenster anzeigt. Sobald diese Taste gedrückt wird, erscheint "MDI CLOSE". Wenn Sie **MDI CLOSE** drücken, wird die MDI geschlossen. Weitere Informationen zur MDI finden Sie im Abschnitt MDI>.
OHMIC-TEST	Mit dieser Taste wird das Ausgangssignal der ohmschen Sonde aktiviert, und wenn der Eingang der ohmschen Sonde erfasst wird, leuchtet die LED-Anzeige im SENSOR-Panel. Der Hauptzweck dieser Taste besteht darin, einen schnellen Test auf eine kurzgeschlossene Brennerspitze zu ermöglichen.
SONDENTEST	Diese Schaltfläche initiiert einen Sondentest.
EINZELSCHNITT (engl. single cut=	Mit dieser Schaltfläche wird das Dialogfeld zum Starten eines automatischen Single Cut angezeigt.
NORMALER SCHNITT	Mit dieser Schaltfläche wird zwischen Schnittarten (engl. cut types) (NORMAL CUT und PIERCE ONLY) umgeschaltet.
BRENNERIMPULS	Diese Taste löst einen Brennerimpuls aus.

Table 63. **ARC**

Name	Modi	Beschreibung
Lichtbogen-Spannung	0, 1	Zeigt die aktuelle Lichtbogenspannung an.
OK	0, 1, 2	Zeigt den Status des Arc OK-Signals an.
+	0, 1	Mit jedem Druck auf diese Taste wird die Zielspannung um die THC-Schwellenspannung erhöht (der geänderte Abstand ergibt sich aus Höhe pro Volt * THC-Schwellenspannung).
-	0, 1	Mit jedem Druck auf diese Taste wird die Zielspannung um die THC-Schwellenspannung gesenkt (der geänderte Abstand ist Höhe pro Volt * THC-Schwellenspannung).
OVERRIDE	0, 1	Wenn Sie auf dieses Etikett klicken, wird eine eventuelle Spannungsübersteuerung auf 0,00 zurückgesetzt.

Table 64. **CONTROL**

Name	Modi	Beschreibung
BRENNER AN (engl. torch on)	0, 1, 2	Zeigt den Status des Ausgangssignals Torch On an.
BRENNER AKTIVIEREN	0, 1, 2	Mit diesem Feld können Sie zwischen der Aktivierung und Deaktivierung des Brenners wechseln. Dieses Feld ist standardmäßig nicht ausgefüllt (deaktiviert), wenn QtPlasmaC zum ersten Mal gestartet wird. Dieses Feld muss ausgefüllt werden, um es in "Brenner aktiviert" zu ändern, bevor mit dem Schneiden von Material begonnen werden kann. Wenn dieses Feld nicht ausgefüllt ist, führt die Ausführung eines geladenen Programms dazu, dass die Maschine den Zyklus ohne Zündung des Brenners ausführt. Dies wird manchmal als "Trockenlauf" (engl. dry run) bezeichnet. Wenn der Benutzer einen Laser installiert hat, ist es auch möglich, einen solchen Testlauf mit dem Laser durchzuführen. Siehe Abschnitt LASER Abschnitt für detaillierte Anweisungen.
VELOCITY ANTI DIVE	0, 1, 2	Zeigt an, dass die THC auf der aktuellen Höhe blockiert ist, weil die Schnittgeschwindigkeit unter den prozentualen Schwellenwert für den Geschwindigkeitsabbau (VAD) fällt, der auf der Registerkarte PARAMETER eingestellt ist.
VELOCITY ANTI DIVE ENABLE	0, 1, 2	Dieses Feld schaltet zwischen Aktivieren (engl. enabling) und Deaktivieren (engl. disabling) von VELOCITY ANTI DIVE um.
VOID ANTI DIVE	0, 1	Zeigt an, dass der THC gesperrt ist, weil ein Leerraum (engl. void) erkannt wurde.
VOID ANTI DIVE ENABLE	0, 1	Dieses Feld schaltet zwischen Aktivieren (engl. enabling) und Deaktivieren (engl. disabling) von VOID ANTI DIVE um.
MESH-MODUS	0, 1, 2	Dieses Kästchen aktiviert oder deaktiviert Mesh Mode für das Schneiden von Streckmetall. Dieses Kontrollkästchen kann während des normalen Schneidens jederzeit aktiviert oder deaktiviert werden. Mesh-Modus: - Erfordert ein Lichtbogen-OK-Signal, um die Maschinenbewegung zu starten. - Deaktiviert die THC. - Stoppt die Maschinenbewegung nicht, wenn das Lichtbogen-OK-Signal verloren geht. - Wählt automatisch den CPA-Modus, wenn die PowerMax-Kommunikation verwendet wird. Für weitere Informationen siehe Mesh Mode (Streckmetall).
AUTO VOLTS	0, 1	Mit diesem Feld wird Auto Volts aktiviert oder deaktiviert.

Name	Modi	Beschreibung
IGNORE OK	0, 1, 2	Dieses Kontrollkästchen bestimmt, ob QtPlasmaC das Signal Arc OK ignoriert. Dieses Kontrollkästchen kann während des normalen Schneidens jederzeit aktiviert oder deaktiviert werden. Zusätzlich kann dieser Modus durch entsprechende M-Codes in einem laufenden Programm aktiviert oder deaktiviert werden. Lichtbogen-OK-Modus ignorieren: - Erfordert nicht, dass ein Lichtbogen-OK-Signal empfangen wird, bevor die Maschinenbewegung nach dem Signal "Brenner ein" gestartet wird. - Deaktiviert die THC. - Stoppt die Maschinenbewegung nicht, wenn das Lichtbogen-OK-Signal verloren geht. Für weitere Informationen siehe Ignore Arc Ok.
OHMIC PROBE	0, 1, 2	Dieses Feld aktiviert oder deaktiviert den Eingang der ohmschen Sonde. Wenn der Eingang der ohmschen Sonde deaktiviert ist, zeigt die LED der ohmschen Sonde weiterhin den Status des Sondeneingangs an, aber die Ergebnisse der ohmschen Sonde werden ignoriert.
RS485	0, 1, 2	Mit diesem Feld wird die Kommunikation mit einem PowerMax aktiviert oder deaktiviert. Diese Schaltfläche ist nur sichtbar, wenn im Abschnitt [POWERMAX] der Datei <*machine_name*>.prefs eine PM_PORT Option konfiguriert ist.
Status	0, 1, 2	Wenn die PowerMax-Kommunikation aktiviert ist, wird hier eine der folgenden Anzeigen erscheinen: **CONNECTING** (engl. für Verbindungsaufbau), **CONNECTED** (engl. für verbunden), **COMMS ERROR** (engl. kurz für Kommunikationsfehler) oder ein **Fehlercode** (engl. fault code). Weitere Informationen finden Sie im Abschnitt PowerMax Communications.

Table 65. **SENSOR**

Name	Beschreibung
FLOAT	Zeigt an, dass der Schwimmerschalter aktiviert ist.
OHMIC	Zeigt an, dass die Sonde das Material abgetastet hat.
ABRISS (engl. break)	Zeigt an, dass der Abreißsensor des Brenners aktiviert ist.

Table 66. **THC**

Name	Beschreibung
ENABLE (AKTIVIEREN)	Dieses Feld bestimmt, ob die THC während eines Schnitts aktiviert oder deaktiviert wird.
AKTIVIERT (engl. enabled)	Diese LED zeigt an, ob die THC aktiviert oder deaktiviert ist.
AKTIV	Diese LED zeigt an, dass die THC die Z-Achse aktiv steuert.
HOCH	Diese LED zeigt an, dass die THC den Befehl zum Anheben der Z-Achse gibt.
HERAB	Diese LED zeigt an, dass die THC den Befehl zum Absenken der Z-Achse gibt.

NOTE

JOGGING

Bei angehaltener Bewegung wird dieser Abschnitt zu CUT RECOVERY

Name	Beschreibung
KONTINUIERLICH	Diese Dropdown-Schaltfläche ändert die Schrittweite des Jogs. Die Optionen werden durch die Werte im Abschnitt **[DISPLAY]** in der Datei <*Maschinenname*>.ini bestimmt und beginnen mit der Bezeichnung "INCREMENTS =".
SCHNELL (engl. fast)	Diese Schaltfläche schaltet zwischen SCHNELL, der Standardgeschwindigkeit in der Datei <*Maschinenname*>.ini, und LANGSAM, die 10 % des Standardwerts beträgt, um.
Y+	Diese Schaltfläche verschiebt die Y-Achse in die positive Richtung.
Y-	Diese Schaltfläche verschiebt die Y-Achse in die negative Richtung.
X+	Mit dieser Schaltfläche wird die X-Achse in die positive Richtung bewegt.
X-	Diese Schaltfläche verschiebt die X-Achse in die negative Richtung.
Z+	Mit dieser Schaltfläche wird die Z-Achse in die positive Richtung bewegt.
Z-	Diese Schaltfläche verschiebt die Z-Achse in die negative Richtung.

NOTE

CUT RECOVERY

Bei angehaltener Bewegung wird dieser Bereich über dem Feld JOGGING angezeigt. Im folgenden Abschnitt wird jede Schaltfläche in diesem Bereich beschrieben. Eine detaillierte Beschreibung der Schnittwiederherstellung finden Sie unter CUT RECOVERY.

Name	Beschreibung
SCHIEBEREGLER FÜR ANGEHALTENE BEWEGUNGSABLÄUFE	Im Falle eines angehaltenen Programms ermöglicht diese Schnittstelle eine X/Y-Bewegung, die dem programmierten Pfad in Rückwärts- oder Vorwärtsrichtung folgt. Der Bereich dieses Schiebereglers liegt zwischen 1 % und 100 % der Schnittvorschubgeschwindigkeit für das aktuell ausgewählte Material.

Name	Beschreibung
FEED	Hier wird die Vorschubgeschwindigkeit der angehaltenen Bewegung angezeigt.
RÜCKW	Im Falle eines angehaltenen Programms bewegt diese Schaltfläche die Maschine rückwärts entlang des programmierten Pfades, bis sie den letzten M3-Befehl erreicht, der entweder ausgeführt wurde oder den QtPlasmaC versucht hat auszuführen, bevor das Programm angehalten wurde.
VORW	Im Falle eines angehaltenen Programms bewegt sich die Maschine mit dieser Taste auf dem programmierten Weg unbegrenzt bis zum Ende des Programms vorwärts, wobei M3-Befehle übersprungen werden.
BEWEGUNG ABBRECHEN (engl. cancel move)	Diese Schaltfläche bricht alle durchgeführten Schnittwiederherstellungsbewegungen ab und bringt den Brenner wieder in die Position, an der die Schnittwiederherstellungsbewegung initiiert wurde. Beachten Sie, dass, wenn FWD oder REV zum Bewegen der Taschenlampe verwendet wurde, CANCEL nicht in die Position des Brenners zurückkehrt, als die Pause aufgetreten ist.
BEWEGEN x.xxx	Hier wird der Weg angezeigt, der bei jedem Drücken einer Pfeiltaste in der Richtung zurückgelegt wird, in der die Pfeiltaste gedrückt wurde. Dieser Wert, der unter BEWEGEN (engl. move) angezeigt wird, stellt die Fugenbreite des aktuell ausgewählten Materials dar.
RICHTUNGSPFEILE	Diese BUTTONS bewegen den Brenner in die angegebene Richtung um eine Kerbbreite (des aktuell ausgewählten Materials) pro Druck.

Table 67. **G-CODE FENSTER**

Name	Beschreibung
CLEAR	Diese Schaltfläche wird das aktuell geöffnete Programm löschen. Wenn eine Datei geöffnet ist, wird das Standardmaterial ausgewählt. Wenn keine Datei geöffnet ist, wird der Preview> auf eine obere nach unten vollständige Tabellenansicht zurückgesetzt. Der Brenner (T0) wird ausgewählt, wenn er nicht das aktive Werkzeug war. Vorherige Fehlermeldungen und der Fehler-Status werden gelöscht. Die Schnittart (engl. cut type) wird gesetzt auf NORMAL CUT.
OFFNEN	Diese Schaltfläche öffnet ein DATEI-ÖFFNEN-Panel über dem VORSCHAU-FENSTER.
NEU LADEN	Diese Schaltfläche lädt die aktuell geladene G-Code-Datei neu.

Table 68. **DRO**

Name	Beschreibung
ALLE REFERENZIEREN	Mit dieser Schaltfläche werden alle Achsen in der durch HOME_SEQUENCE in der Datei <*Maschinenname*>.ini festgelegten Reihenfolge ausgerichtet.

Name	Beschreibung
WCS G54	Mit dieser Dropdown-Schaltfläche können Sie den aktuellen Arbeitsversatz ändern.
KAMERA	Diese Schaltfläche zeigt ein CAMVIEW-Panel über dem VORSCHAU-FENSTER an und ermöglicht es dem Benutzer, einen Ursprung mit oder ohne Drehung festzulegen. Siehe den Abschnitt zur KAMERA für detaillierte Anweisungen. Dieser Button wird nich sichtbar bis ein KAMERA Offset in der <*Maschinenname*>.prefs-Datei gesetzt wurde.
LASER	Mit dieser Schaltfläche kann der Benutzer einen Laser verwenden, um einen Ursprung mit oder ohne Drehung festzulegen. Siehe den Abschnitt LASER für detaillierte Anweisungen. Dieser Button wird nicht sichtbar bis ein LASER Offset gedsetzt ist in der <*Maschinenname*>.prefs Datei.
X0 Y0	Mit dieser Schaltfläche wird die aktuelle Position auf X0 Y0 gesetzt.
HOME [AXIS]	Mit dieser Schaltfläche wird die entsprechende Achse referenziert.
0 [ACHSE]	Diese Dropdown-Schaltfläche zeigt die folgenden Optionen an: **Null** - setzt die Achse auf Null. **Setzen** - öffnet ein Dialogfeld zur manuellen Eingabe der Achsenkoordinaten. **Teile durch 2** - teilt die aktuell angezeigte Koordinate im DRO durch zwei. **Setze auf vorherige** - setzt die Achse auf die zuvor eingestellte Koordinate.

Vorschau-Ansichten

Der Vorschaubildschirm von QtPlasmaC kann zwischen verschiedenen Ansichten und Displays umgeschaltet werden, und man kann hinein- und herauszoomen sowie horizontal und vertikal schwenken.

Wenn QtPlasmaC zum ersten Mal gestartet wird, dann wird die Z-Ansicht (von oben nach unten) als Standardansicht für eine geladene G-Code-Datei ausgewählt, aber die vollständige Tabellenansicht wird angezeigt.

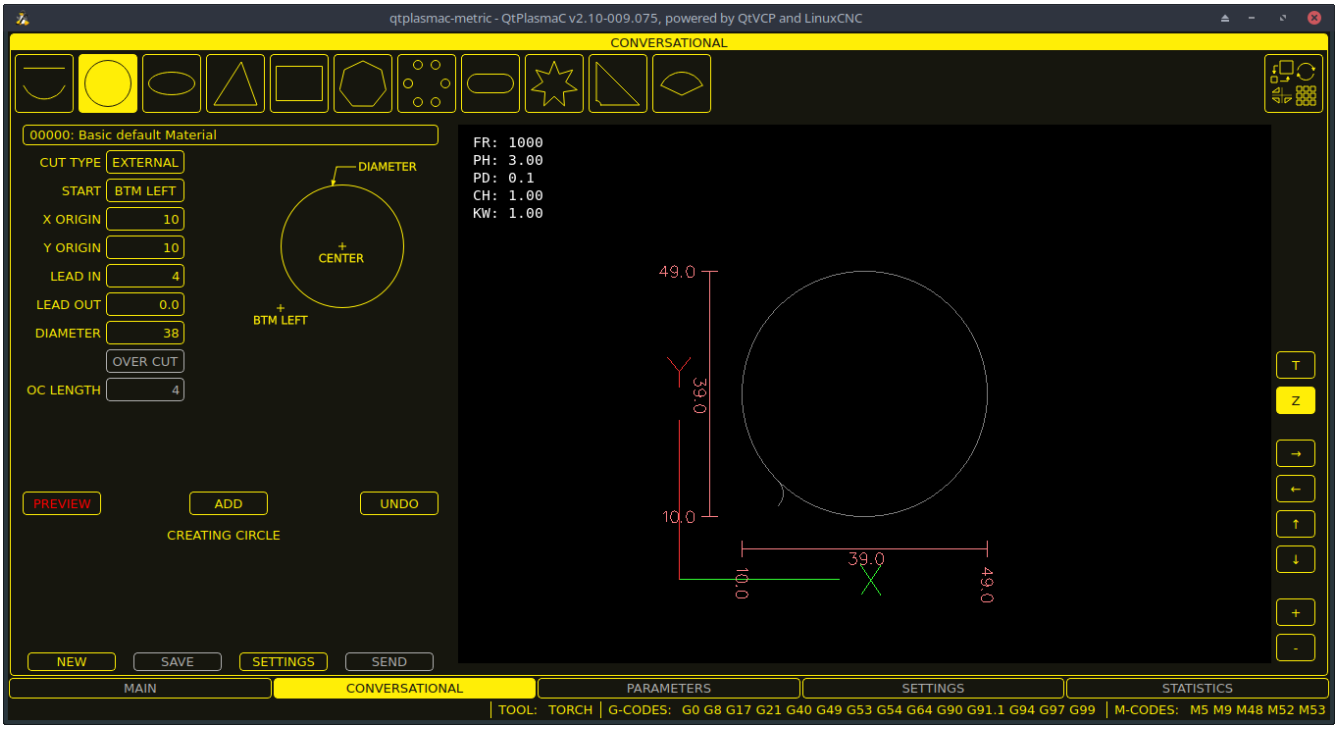
Wenn eine G-Code-Datei geladen wird, wechselt die Anzeige zur ausgewählten Ansicht.

Wenn keine G-Code-Datei geladen ist, wird automatisch die vollständige Tabelle angezeigt, unabhängig davon, welche Ansicht gerade ausgewählt ist (die hervorgehobene Schaltfläche, welche die gerade ausgewählte Ansicht darstellt, ändert sich nicht).

Wenn eine vollständige Tabelle angezeigt wird, weil keine G-Code-Datei geladen ist, und der Benutzer die Ansichtsausrichtung ändern möchte, wird durch Drücken von Z oder P die Anzeige auf die neu gewählte Ansicht umgestellt. Wenn der Benutzer dann die vollständige Tabelle anzeigen möchte, während er die aktuell gewählte Ansicht als Standardansicht für eine geladene G-Code-Datei beibehält, kann er dies durch Drücken von CLEAR erreichen und die gewählte Ansichtsausrichtung beim nächsten Laden einer G-Code-Datei beibehalten.

Registerkarte "KONVERSATION"

Screenshot-Beispiel der QtPlasmaC-Registerkarte CONVERSATIONAL im Seitenverhältnis **16:9**:



Die Registerkarte CONVERSATIONAL ermöglicht es dem Benutzer, verschiedene einfache Formen zum schnellen Schneiden zu programmieren, ohne dass eine CAM-Software erforderlich ist.

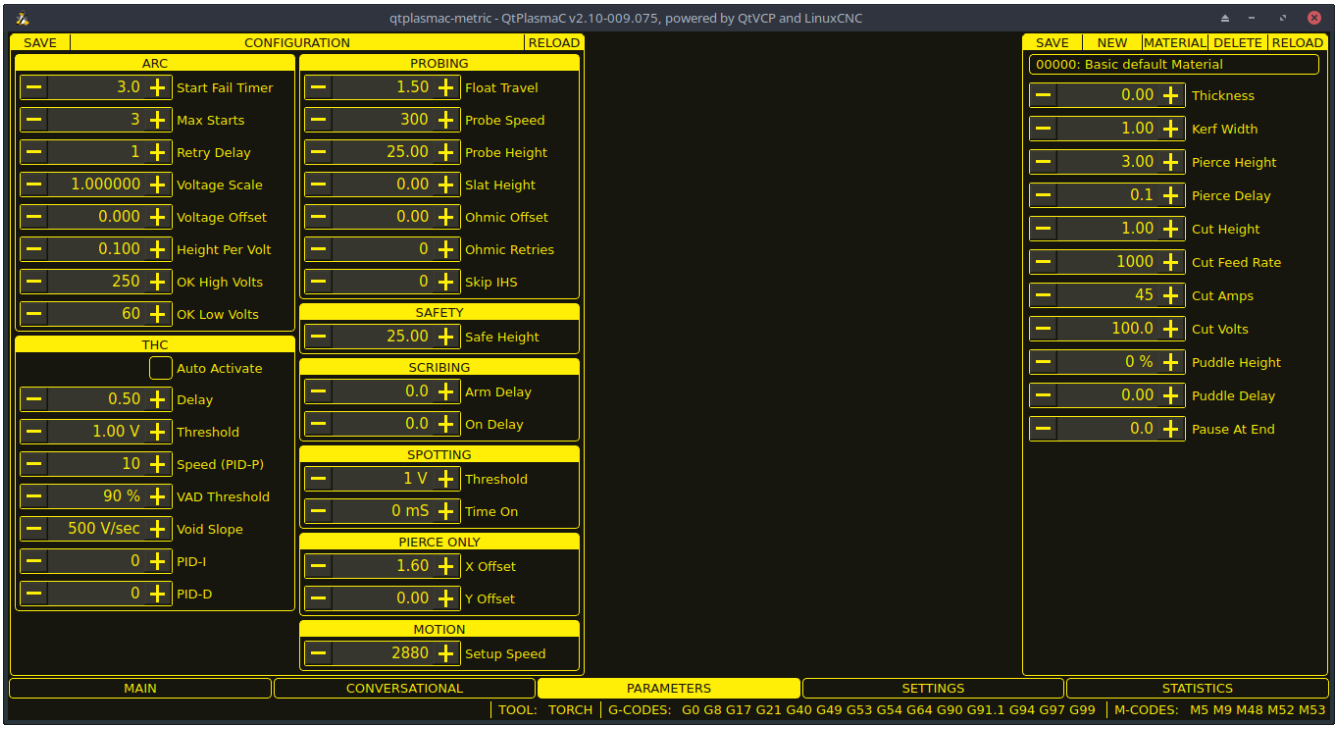
Siehe Conversational Shape Library für detaillierte Informationen über die Conversational-Funktion.

Es ist möglich, diese Registerkarte zu verstecken, damit die Konversationsfunktion nicht von einem Bediener verwendet werden kann. Dies kann entweder durch Verdrahtung des Pins mit einem physischen Schlüsselschalter oder ähnlichem erreicht werden, oder es kann auch in einer HAL-Datei mit dem folgenden Befehl eingestellt werden:

```
setp qtplasmac.conv_disable 1
```

Registerkarte PARAMETER

Screenshot-Beispiel des QtPlasmaC-Registerkarte PARAMETER im Seitenverhältnis **16:9**:



Einige Funktionen/Merkmale werden nur für bestimmte Modi verwendet und werden nicht angezeigt, wenn sie für den gewählten QtPlasmaC-Modus nicht erforderlich sind.

Diese Registerkarte dient zur Anzeige von Konfigurationsparametern, die nur selten geändert werden.

Es ist möglich, diese Registerkarte zu verstecken, damit die Maschineneinstellungen nicht von unbefugtem Personal geändert werden können. Dies kann entweder durch Verdrahtung des Pins mit einem physischen Schlüsselschalter oder ähnlichem erreicht werden, oder es kann auch in einer HAL-Datei mit dem folgenden Befehl eingestellt werden:

```
setp qtplasmac.param_disable 1
```

Table 69. **KONFIGURATION - ARC**

Name	Modi	Beschreibung
Start-Ausfall-Timer	0, 1, 2	Hier wird die Zeitspanne (in Sekunden) festgelegt, die QtPlasmaC zwischen dem Befehl "Brenner ein" und dem Empfang eines Lichtbogen-OK-Signals wartet, bevor die Zeitspanne abgelaufen ist und eine Fehlermeldung angezeigt wird.
Max. Starts	0, 1, 2	Hier wird festgelegt, wie oft QtPlasmaC versuchen soll, den Lichtbogen zu starten.
Wiederholungsverzögerung	0, 1, 2	Hier wird die Zeit (in Sekunden) zwischen einem fehlgeschlagenen Lichtbogen und einem erneuten Startversuch eingestellt.
Spannungsskala	0, 1	Hier wird die Eingangsskala für die Lichtbogenspannung eingestellt und die korrekte Lichtbogenspannung angezeigt. Zur Ersteinrichtung siehe Kalibrations-Werte.

Name	Modi	Beschreibung
Spannungs-Offset	0, 1	Hier wird der Offset für die Lichtbogenspannung eingestellt. Er wird verwendet, um Null Volt anzuzeigen, wenn keine Lichtbogenspannung anliegt. Für die Ersteinrichtung siehe Kalibrations-Werte.
Höhe pro Volt	0, 1, 2	Damit wird die Entfernung festgelegt, die der Brenner zurücklegen muss, um die Lichtbogenspannung um ein Volt zu ändern. Wird nur für die manuelle Höhenverstellung verwendet.
OK Hohe Spannungen	0	Damit wird die Spannungsschwelle festgelegt, unterhalb derer das Signal Arc OK gültig ist.
OK Niedrige Spannung [Volt]	0	Hier wird die Spannungsschwelle festgelegt, ab der das Lichtbogen-OK-Signal gültig ist.

NOTE Bei der Einstellung von OK Low Volts und OK High Volts in Modus 0 muss die Schaltspannung eines stabilen Lichtbogens größer als der Wert für OK Low Volts, aber kleiner als der Wert für OK High Volts sein, damit QtPlasmaC ein gültiges Lichtbogen-OK-Signal erhält. Zur weiteren Verdeutlichung: Um ein gültiges Lichtbogen-OK zu erhalten, muss die Lichtbogenspannung zwischen diesen beiden Grenzwerten liegen.

Table 70. **KONFIGURATION - SONDIERUNG**

Name	Beschreibung
Tasterweg (engl. Float Travel)	Dies legt den Tasterweg fest, den der Float-Schalter zurücklegt, bevor der Schalterkreis geschlossen wird. Dieser Abstand kann mithilfe der Schaltfläche Sondentest und der unter Ersteinrichtung beschriebenen Methode gemessen werden.
Sonden-Geschwindigkeit	Hier wird die Geschwindigkeit eingestellt, mit welcher der Brenner nach Erreichen der Sondenhöhe nach dem Material suchen soll.
Sondenhöhe	This sets the height above the Z axis minimum limit that Probe Speed begins. If set to zero, then the torch will move at Probe Speed from the current position. This may be adventageous when using Slat Height and Material Thickness as the machine will default to Probe Height if either Slat Height or Material Thickness are zero. Refer to the Heights Diagrams for a visual representation.
Slat Height	This sets the height of the slats, measured up from Z axis minimum limit. This must be used in conjunction with Material Thickness. If either Slat Height or Material Thickness are zero then the machine will default to Probe Height. Refer to the Heights Diagrams for a visual representation.
Ohmscher Offset	Damit wird der Abstand über dem Material festgelegt, den der Brenner nach einer erfolgreichen ohmschen Sonde erreichen soll. Er wird hauptsächlich verwendet, um hohe Antastgeschwindigkeiten zu kompensieren.

Name	Beschreibung
Ohmsche Wiederholungen	Hier wird festgelegt, wie oft QtPlasmaC einen fehlgeschlagenen ohmschen Messfühler erneut versucht, bevor es zur Materialerkennung auf den Schwimmerschalter zurückgreift.
IHS überspringen	Hier wird der Abstandsschwellenwert festgelegt, der verwendet wird, um zu bestimmen, ob eine anfängliche Höhenabtastung (Probe) für den aktuellen Schnitt übersprungen werden kann, siehe IHS Überspringen.
Offset-Geschwindigkeit	Hier wird die Geschwindigkeit eingestellt, mit der sich die Sonde in der X-Achse und Y-Achse in die Offsetposition bewegt.

NOTE Wenn die Zeitspanne zwischen dem Kontakt des Brenners mit dem Material und dem Zeitpunkt, zu dem der Brenner sich nach oben bewegt und auf der Durchstechhöhe zur Ruhe kommt, zu lang erscheint, finden Sie im << plasma:probing,Abschnitt Antasten>> eine mögliche Lösung.

Table 71. **KONFIGURATION - SICHERHEIT**

Name	Beschreibung
Sichere Höhe	This sets the height above the material that the torch will retract to before executing rapid moves. If set to zero, then Max Offset (plasmac.max-offset) will be used for the safe height. This defaults to 5mm (0.197"). Refer to the Heights Diagrams for a visual representation.

Table 72. **KONFIGURATION - GRAVIEREN** *(engl. scribing)*

Name	Beschreibung
Arm-Verzögerung	Hier wird die Verzögerung (in Sekunden) zwischen dem Empfang des Gravier (engl. scribe)-Befehls und der Aktivierung des Gravierens eingestellt. Dies ermöglicht es dem Gravierer, die Oberfläche des Materials zu erreichen, bevor der Gravierer aktiviert wird.
Eingangs-Verzögerung	Legt die Verzögerung (in Sekunden) fest, mit der Ritzmechanismus gestartet wird, bevor die Bewegung beginnt.

Table 73. **KONFIGURATION - SPOTTING**

Name	Beschreibung
Schwellenwert	Dies stellt die Lichtbogenspannung ein, ab welcher der Verzögerungszeitgeber beginnt. 0 V startet die Verzögerung, wenn das Brenner-Ein-Signal aktiviert wird.
Laufzeit (engl. Time On)	Hier wird die Zeitspanne (in Millisekunden) festgelegt, die der Brenner nach Erreichen der Schwellenspannung eingeschaltet ist.

Table 74. **CONFIGURATION - PIERCE ONLY**

Name	Beschreibung
X Versatz (engl. offset)	Bewegt den Durchstech (engl. pierce)-Punkt diesen Abstand entlang der X-Achse beim Durchstechen (engl. piercing) im Pierce Only Modus.
Y Versatz (engl. offset)	Bewegt den Durchstech (engl. pierce)-Punkt diesen Abstand entlang der Y-Achse beim Durchstechen (engl. piercing) im Pierce Only Modus.

Table 75. **KONFIGURATION - BEWEGUNG** *(engl. motion)*

Name	Beschreibung
Einstellung Geschwindigkeit	Die Geschwindigkeit der Z-Achse für Einrichtungsbewegungen (Bewegungen zur Sondenhöhe, Lochstechhöhe, Schnitthöhe usw.).

NOTE Die Setup-Geschwindigkeit hat keinen Einfluss auf die THC-Geschwindigkeit, die in der Lage ist, die im Feld Max. Speed angezeigte Geschwindigkeit zu erreichen.

Table 76. **KONFIGURATION - THC**

Name	Modi	Beschreibung
Delay (engl. für Verzögerung)	0, 1, 2	Hier wird die Verzögerung (in Sekunden) eingestellt, die vom Empfang des Lichtbogen-OK-Signals bis zur Aktivierung der Brennerhöhensteuerung (THC) gemessen wird. Dies ist nur verfügbar, wenn Auto THC nicht aktiviert ist.
Anzahl der Stichproben	0, 1	Hier wird die Anzahl der aufeinanderfolgenden Lichtbogen-Spannungsmessungen innerhalb der THC-Probenschwelle eingestellt, die zur Aktivierung der Brennerhöhensteuerung (THC) erforderlich ist. Dies ist nur verfügbar, wenn Auto THC aktiviert ist.
Proben Schwellwert	0, 1	Hier wird die maximal zulässige Spannungsabweichung für THC-Probenzählungen eingestellt. Dies ist nur verfügbar, wenn Auto THC aktiviert ist.
Schwellenwert	0, 1	Damit wird die zulässige Spannungsabweichung von der Sollspannung festgelegt, bevor die THC Bewegungen zur Korrektur der Brennerhöhe ausführt.
Geschwindigkeit (PID-P)	0, 1, 2	Dies legt die proportionale Verstärkung für die THC-PID-Schleife fest. Dies entspricht in etwa der Geschwindigkeit, mit der das THC versucht, Höhenänderungen zu korrigieren.
VAD-Schwellenwert	0, 1, 2	(Velocity Anti Dive) Hier wird der Prozentsatz des aktuellen Schnittvorschubs eingestellt, auf den die Maschine verlangsamen kann, bevor die THC gesperrt wird, um ein Eintauchen des Brenners zu verhindern.

Name	Modi	Beschreibung
Void (engl. für Leerraum)-Neigung	0, 1	(Void Anti Dive) Hier wird die Größe der Änderung der Abschaltspannung pro Sekunde eingestellt, die erforderlich ist, um die THC zu verriegeln, um ein Absinken des Brenners zu verhindern (höhere Werte erfordern eine größere Spannungsänderung, um die THC zu verriegeln).
PID-I	0, 1	Dies stellt die Integralverstärkung für die THC-PID-Schleife ein. Die integrale Verstärkung ist mit der Summe der Fehler im System über die Zeit verbunden und wird nicht immer benötigt.
PID-D	0, 1	Dies legt den abgeleiteten Gewinn für die THC-PID-Schleife fest. Die abgeleitete Verstärkung dämpft das System und reduziert Überkorrekturschwingungen und ist nicht immer erforderlich.

Es stehen zwei Methoden der THC-Aktivierung zur Verfügung, die über den Check-button **Auto Activation** ausgewählt werden. Beide Methoden beginnen ihre Berechnungen, wenn die aktuelle Geschwindigkeit des Brenners mit der für das ausgewählte Material festgelegten Schnittvorschubgeschwindigkeit übereinstimmt:

1. Die verzögerte Aktivierung (die Standardeinstellung) wird ausgewählt, wenn **Automatische Aktivierung** nicht markiert ist. Diese Methode verwendet eine Zeitverzögerung, die mit dem Parameter **Delay** eingestellt wird.
2. Die automatische Aktivierung wird ausgewählt, wenn **Auto Activation** markiert ist. Diese Methode bestimmt, dass die Lichtbogenspannung stabil ist, indem sie die Parameter **Sample Counts** und **Sample Threshold** verwendet.

NOTE Die Abstimmung von PID-Regelkreisen ist ein komplizierter Prozess und liegt außerhalb des Rahmens dieses Benutzerhandbuchs. Es gibt viele Informationsquellen, die das Verständnis und die Abstimmung von PID-Regelkreisen unterstützen. Wenn die THC Korrekturen nicht schnell genug vornimmt, wird empfohlen, die P-Verstärkung in kleinen Schritten zu erhöhen, bis das System günstig arbeitet. Große P-Verstärkungseinstellungen können zu einer Überkorrektur und zu Schwingungen führen.

Speichern *(engl. save)* **NEU LADEN** *(engl. reload)* **Schaltflächen** *(engl. buttons)*

Die Schaltfläche **SPEICHERN** speichert die aktuell angezeigten Parameter in der Datei <*Maschinenname*>.prefs.

Mit der Schaltfläche **RELOAD** werden alle Parameter aus der Datei <*Maschinenname*>.prefs neu geladen.

Table 77. **MATERIAL** *- Die für den aktuellen Schnitt aktiven Parameter.*

Name	Beschreibung
Material	Das obere Dropdown-Menü dient zur manuellen Auswahl der aktuellen Materialschnittparameter. Wenn in der Materialdatei keine Materialien vorhanden sind, wird nur das Standardmaterial angezeigt.

Name	Beschreibung
Thickness	This sets the thickness for the currently selected material. This must be used in conjunction with Slat Height. If either Slat Height or Material Thickness are zero then the machine will default to Probe Height. Refer to the Heights Diagrams for a visual representation.
Schnittfugenbreite (engl. kerf width)	Hiermit wird die Schnittfugenbreite (engl. kerf width) für das aktuell ausgewählte Material festgelegt. Eine visuelle Darstellung finden Sie im Höhen Diagramm (engl. Heights Diagram).
Einstichhöhe (engl. pierce height)	Hiermit wird die Durchstoßhöhe (engl. widthpierce height) für das aktuell ausgewählte Material festgelegt. Eine visuelle Darstellung finden Sie im Höhen Diagramm (engl. Heights Diagram).
Durchstichverzögerung (engl. pierce delay)	Damit wird der pierce delay (in Sekunden) für das aktuell ausgewählte Material eingestellt.
Schnitthöhe (engl. cut height)	Hiermit wird die Schnitthöhe (engl. cut height) für das aktuell ausgewählte Material festgelegt. Eine visuelle Darstellung finden Sie im Höhen Diagramm (engl. Heights Diagram).
Schnitt Vorschubgeschwindigkeit	Hier wird der Schnittvorschub für das aktuell ausgewählte Material eingestellt.
Schnitt-Stromstärke (engl. cut amps)	Hiermit wird die Stromstärke für das aktuell ausgewählte Material eingestellt. Dies ist nur eine visuelle Anzeige für den Bediener, es sei denn, die PowerMax-Kommunikation wird verwendet.
Schnittspannungen (engl. cut volts)	Hiermit wird die Schnittspannung für das aktuell ausgewählte Material eingestellt.
Pfützenhöhe (engl. puddle height)	Ausgedrückt als Prozentsatz der Pierce-Höhe legt dies die Pfützensprunghöhe (engl. puddle jump height) für das aktuell ausgewählte Material fest. Puddle Jump wird normalerweise für dickere Materialien verwendet und ermöglicht es dem Brenner, einen Zwischenschritt zwischen Durchstichhöhe und Schnitthöhe zu haben. Wenn eingestellt, wird der Brenner für einen bestimmten Zeitraum von der Durchstichhöhe zur P-Sprunghöhe (P-Sprungverzögerung) fortgesetzt, bevor sie zur Schnitthöhe übergeht, um effektiv über die geschmolzene Pfütze zu "springen". Eine visuelle Darstellung finden Sie im Höhendiagramm.
Pfützenverzögerung (engl. puddle delay)	Damit wird die Zeitdauer (in Sekunden) eingestellt, die der Brenner auf der P-Sprunghöhe (engl. P-Jump Height) bleibt, bevor er zur Schnitthöhe (engl. cut height) übergeht.
Pause am Ende	Damit wird die Zeitdauer (in Sekunden) eingestellt, die der Brenner am Ende des Schnitts eingeschaltet bleibt, bevor mit dem M5-Befehl zum Ausschalten und Anheben des Brenners fortgefahren wird. Weitere Informationen finden Sie unter Pause am Ende des Schnitts.

Name	Beschreibung
Gasdruck	Hier wird der Gasdruck für das aktuell ausgewählte Material eingestellt. Diese Einstellung ist nur gültig, wenn die PowerMax-Kommunikation verwendet wird. 0 = Verwenden Sie den automatischen Druckmodus des PowerMax.
Schnittmodus (engl. cut mode)	Hiermit wird der Schneidemodus für das aktuell ausgewählte Material eingestellt. Diese Einstellung ist nur gültig, wenn die PowerMax-Kommunikation verwendet wird. 1 = Normal 2 = CPA (Konstanter Pilotbogen) 3 = Schauglas (engl. gouge)/Markierung

NOTE Weitere Informationen zum Pfützensprung finden Sie im Abschnitt thick materials.

Buttons für Speichern, Neu laden, Neu und Löschen

Mit der Schaltfläche **SPEICHERN** wird der aktuelle Materialsatz in der Datei <*Maschinenname*>_material.cfg gespeichert.

Mit der Schaltfläche **NEU LADEN** (engl. reload) wird der Materialsatz aus der Datei <*Maschinenname*>_material.cfg neu geladen.

Mit der Schaltfläche **NEU** kann ein neues Material zur Materialdatei hinzugefügt werden. Der Benutzer wird zur Eingabe einer Materialnummer und eines Materialnamens aufgefordert, alle anderen Parameter werden aus dem aktuell ausgewählten Material gelesen. Nach der Eingabe lädt QtPlasmaC die Materialdatei neu und zeigt das neue Material an. Die Schnittparameter für das neue Material müssen dann angepasst und gespeichert werden.

Die Schaltfläche **LÖSCHEN** wird zum Löschen eines Materials verwendet. Nach dem Drücken dieser Taste wird der Benutzer aufgefordert, eine Materialnummer einzugeben, die gelöscht werden soll, und er wird erneut gefragt, um sicherzustellen, dass er sich sicher ist. Nach dem Löschen wird die Materialdatei neu geladen und in der Dropdown-Liste wird das Standardmaterial angezeigt.

Registerkarte EINSTELLUNGEN

Screenshot-Beispiel des QtPlasmaC-Registerkarte EINSTELLUNGEN im Seitenverhältnis **16:9**:



Diese Registerkarte wird verwendet, um GUI-Konfigurationsparameter, Schaltflächentext und Herunterfahrtext anzuzeigen, die selten geändert werden, sowie einige Dienstprogrammschaltflächen.

Es ist möglich, diese Registerkarte zu verstecken, damit die Maschineneinstellungen nicht von unbefugtem Personal geändert werden können. Dies kann entweder durch Verdrahtung des Pins mit einem physischen Schlüsselschalter oder ähnlichem erreicht werden, oder es kann auch in einer HAL-Datei mit dem folgenden Befehl eingestellt werden:

```
setp qtplasmac.settings_disable 1
```

GUI EINSTELLUNGEN *(engl. settings)*

Dieser Abschnitt zeigt Parameter, die das Erscheinungsbild und das Verhalten der grafischen Benutzeroberfläche (GUI) beeinflussen.

Informationen zum Zurücksetzen von Farbänderungen auf ihre Standardwerte finden Sie im Abschnitt Zurück zum Standardstil.

Table 78. **GUI EINSTELLUNGEN** *Parameter, die das Erscheinungsbild und das Verhalten der grafischen Benutzeroberfläche (GUI) beeinflussen.*

Name	Beschreibung
Vordergrund	Mit dieser Schaltfläche kann der Benutzer die Farbe des GUI-Vordergrunds ändern.
Hervorhebung	Mit dieser Schaltfläche kann der Benutzer die Farbe des GUI-Highlights ändern.
LED	Mit dieser Schaltfläche kann der Benutzer die Farbe der GUI-LED ändern.
Hintergrund	Mit dieser Schaltfläche kann der Benutzer die Farbe des GUI-Hintergrunds ändern.

Name	Beschreibung
Alternativer Hintergrund	Mit dieser Schaltfläche kann der Benutzer die Farbe des alternativen Hintergrunds der GUI ändern.
Rahmen	Mit dieser Schaltfläche kann der Benutzer die Farbe der GUI-Rahmen ändern.
Notaus	Mit dieser Schaltfläche kann der Benutzer die Farbe der GUI Notaus (engl. E-stop) ändern.
Deaktiviert	Mit dieser Schaltfläche kann der Benutzer die Farbe der deaktivierten Funktionen der grafischen Benutzeroberfläche ändern.
Vorschau	Mit dieser Schaltfläche kann der Benutzer die Farbe des Hintergrunds des GUI-Vorschaufensters ändern.
Soft-Tastatur	Mit dieser Optionsschaltfläche kann der Benutzer die Soft-Touchscreen-Tastatur aktivieren oder deaktivieren. Wenn die "onboard" virtuelle Tastatur installiert ist, wird custom layouts aktiviert.
Tastaturkürzel	Mit diesem Optionsfeld kann der Benutzer Tastaturkürzel innerhalb der GUI aktivieren oder deaktivieren (z.B. Tastatur-Jogging). Zusätzlich zu den Standard-Jogging-Tasten ist eine Liste der zusätzlichen Shortcuts im Abschnitt Tastaturkürzel verfügbar.
Material anzeigen	Mit diesem Optionsfeld kann der Benutzer das Hinzufügen eines visuellen Hinweises mit den wichtigsten Materialschnitt-Einstellungen zu den Vorschaufenstern der Registerkarten MAIN und CONVERSATIONAL aktivieren oder deaktivieren. Beispiele sind: Vorschubgeschwindigkeit, Lochstechhöhe, Lochstechverzögerung und Schnitthöhe. Wenn die PowerMax-Kommunikation aktiviert ist, wird die Stromstärke für den Schnitt angezeigt.
Exit-Warnung	Mit diesem Optionsfeld kann der Benutzer festlegen, ob beim Herunterfahren immer eine Warnung angezeigt werden soll oder nicht. Es ist möglich, der Warnung eine benutzerdefinierte Nachricht hinzuzufügen, indem man die EXIT WARNING MESSAGE-Option im **[GUI_OPTIONS]** Abschnitt der Datei <*machine_name*>.prefs bearbeitet. Die benutzerdefinierte Nachricht kann durch Hinzufügen eines "\" zwischen den Zeilen mehrzeilig gemacht werden.
Optionaler Halt	Mit diesem Optionsfeld kann der Benutzer festlegen, ob ein laufendes Programm bei einem **M1**-Befehl angehalten werden soll oder nicht.
Von Zeile ausführen	Mit diesem Optionsfeld kann der Benutzer Von Zeile Ausführen aktivieren oder deaktivieren. Wenn es aktiviert ist, kann der Benutzer auf eine G-Code-Zeile klicken und das Programm ab dieser Zeile starten lassen.

Name	Beschreibung
Grenzwerte überschreiten	Diese Optionsschaltfläche (engl. radiobutton) ermöglicht es dem Benutzer, den Eingang eines Endschalters vorübergehend außer Kraft zu setzen, falls der Endschalter während des Betriebs ausgelöst wird. Diese Schaltfläche kann nur angeklickt werden, wenn ein Endschalter ausgelöst wurde.
Override Schnellauf (engl. Jog)	Diese Optionsschaltfläche ermöglicht auch das Joggen, wenn das Joggen aufgrund eines Schwimmerschalters, eines Abreißschalters oder der Aktivierung einer ohmschen Sonde gesperrt ist. Diese Schaltfläche kann nur angeklickt werden, wenn das Joggen gesperrt ist.
Optionaler Block	Mit diesem Optionsfeld kann der Benutzer festlegen, ob Zeilen, die mit "/" beginnen, übersprungen werden sollen, wenn sie in einem laufenden Programm vorhanden sind oder nicht.
Rastergröß	Auf diese Weise kann ein Benutzer die Größe des Rasters im Vorschaufenster auf der Registerkarte MAIN ändern. Bei einer Rastergröße von 0,0 wird das Raster deaktiviert.
Kegelgröße	Auf diese Weise kann ein Benutzer die Größe des Kegels (der das aktuelle Werkzeug darstellt) im Vorschaufenster auf der Registerkarte MAIN ändern.
Tabellen-Zoom	Damit kann der Benutzer die Standard-Zoomstufe für die vollständige Tabellenansicht von oben nach unten im Vorschaufenster auf der Registerkarte MAIN ändern.

BENUTZER-BUTTON-EINGABEN *USERBUTTON*

Dieser Abschnitt zeigt den Text, der auf den Custom User Buttons erscheint, sowie den Code, der mit der Benutzerschaltfläche verbunden ist. Die Benutzertasten können geändert und die neuen Einstellungen verwendet werden, ohne dass LinuxCNC neu gestartet werden muss.

Der Text und/oder der Code kann jederzeit bearbeitet werden und wird durch Anklicken der Schaltfläche **SPEICHERN** zur Verwendung geladen.

Wenn Sie den Text **Name** und **Code** löschen, wird die Benutzerschaltfläche ausgeblendet, wenn Sie auf die Schaltfläche **SPEICHERN** klicken.

Um den gesamten Text **Name** und **Code** auf die zuletzt gespeicherten Werte zurückzusetzen, drücken Sie die Taste **RELOAD**.

Name	Code
Der Text, der auf der Schaltfläche angezeigt wird	Der Code, der ausgeführt wird, wenn der Button gedrückt wird.

NOTE Es stehen 20 Benutzerschaltflächen zur Verfügung, die jedoch je nach Fenstergröße nicht alle angezeigt werden können.

EXIT-WARNMELDUNG

Dieser Abschnitt zeigt den Text, der im Dialogfeld zum Herunterfahren angezeigt wird, wenn die Option **Exit-Warnung** aktiviert ist.

Der Text kann jederzeit bearbeitet werden und wird durch Anklicken der Schaltfläche **SPEICHERN** zur Verwendung geladen.

Um den Text **EXIT WARNING MESSAGE** auf den zuletzt gespeicherten Wert zurückzusetzen, drücken Sie die Taste **RELOAD**.

DIENSTPROGRAMME

Einige Standard-LinuxCNC-Dienstprogramme sind als Hilfe bei der Diagnose von Problemen, die auftreten können, zur Verfügung gestellt:

- Halshow
- Halscope
- Halmeter
- Kalibrierung
- Status

Zusätzlich werden die folgenden zwei QtPlasmaC-spezifischen Dienstprogramme bereitgestellt:

Die Schaltfläche **OFFSETS SETZEN** wird verwendet, wenn der Tisch mit einem Laser oder einer Kamera zur Bogenausrichtung (engl. sheet alignment), einem Ritzgerät (engl. scribe) oder einem Offset-Taster ausgestattet ist. Die erforderlichen Offsets für diese Peripheriegeräte müssen nach dem unter << peripheral-offsets, Peripherie-Offsets>> beschriebenen Verfahren angewendet werden.

Die Schaltfläche **KONFIG SICHERN** erstellt eine vollständige Sicherung der Maschinenkonfiguration für die Archivierung oder zur Unterstützung der Fehlerdiagnose. Eine komprimierte Sicherung der Maschinenkonfiguration wird im Linux-Home-Verzeichnis des Benutzers gespeichert. Der Dateiname lautet <*Maschinenname*><*Version*><Datum>_<Uhrzeit>.tar.gz, wobei <*Maschinenname*> der im Konfigurationsassistenten eingegebene Maschinenname ist, <*Version*> die aktuelle QtPlasmaC-Version, die der Benutzer verwendet, <*Datum*> das aktuelle Datum (JJ-MM-TT) und <*Uhrzeit*> die aktuelle Uhrzeit (HH-MM-SS).

Vor der Erstellung des Backups wird das Maschinenprotokoll in einer Datei im Konfigurationsverzeichnis mit dem Namen machine_log_<date>_<time>.txt gespeichert, wobei <*date*> und <*time*> wie oben beschrieben formatiert sind. Diese Datei sowie bis zu fünf frühere Maschinenprotokolle werden ebenfalls in die Sicherung aufgenommen.

Diese Dateien werden von QtPlasmaC nicht benötigt und können jederzeit sicher gelöscht werden.

Registerkarte STATISTIK

Die Registerkarte STATISTICS Tab bietet Statistiken, die es ermöglichen, die Abnutzung der Verbrauchsmaterialien und die Laufzeiten der Aufträge zu verfolgen. Diese Statistiken werden sowohl für den aktuellen Auftrag als auch für die laufende Gesamtzeit angezeigt. Die Statistiken früherer Aufträge werden zurückgesetzt, sobald das nächste Programm ausgeführt wird. Die Gesamtwerte

können entweder einzeln zurückgesetzt werden, indem Sie auf die entsprechende Schaltfläche "RESET" klicken, oder sie können alle zusammen zurückgesetzt werden, indem Sie auf "RESET ALL" klicken.

Das Feld **RS485 PMX STATISTIK** wird nur angezeigt, wenn der Benutzer über eine Hypertherm PowerMax-Kommunikation verfügt und eine gültige RS485-Verbindung mit dem PowerMax hergestellt wurde. Dieses Feld zeigt die **ARC ON TIME** für den PowerMax im Format hh:mm:ss an.

Das **MACHINE LOG** wird auch auf der Registerkarte zu STATISTIKEN angezeigt, dieses Protokoll zeigt alle Fehler und / oder wichtigen Informationen an, die während der aktuellen LinuxCNC-Sitzung auftreten. Wenn der Benutzer eine Sicherungskopie der Konfiguration über die Registerkarte mit EINSTELLUNGEN erstellt, wird auch das Maschinenprotokoll in die Sicherung einbezogen.



10.8.9. QtPlasmaC verwenden

Nach einer erfolgreichen Installation von QtPlasmaC ist keine Z-Achsen-Bewegung erforderlich, um Teil des G-Code-Schneideprogramms zu sein. Falls Z-Achsen-Referenzen im Schnittprogramm vorhanden sind, werden sie von der Standardkonfiguration von QtPlasmaC während des Programmladevorgangs entfernt.

Für eine zuverlässige Verwendung von QtPlasmaC sollte der Benutzer **NICHT** andere Z-Achsen-Offsets als die Koordinatensystem-Offsets (G54-G59.3) verwenden. Aus diesem Grund wurden G92 Offsets im gesamten GUI deaktiviert (engl. disabled).

QtPlasmaC fügt am Anfang jedes G-Code-Programms automatisch eine Zeile G-Code ein, um die Z-Achse auf die richtige Höhe zu bringen.

NOTE Es ist möglich, die Z-Bewegung für die Verwendung mit unterschiedlichen Werkzeugen beizubehalten, indem der magische Kommentar #<keep-z-motion>=1 hinzugefügt wird. Wird eine rotatorische A-, B- oder C-Achse für Rohrschneiden verwendet, ist eine Z-Achsenbewegung im G-Code-Programm erforderlich.

Versionsinformationen - QtPlasmaC zeigt im Titel des Hauptfensters Informationen zur Versionierung an. Die Informationen werden wie folgt angezeigt "QtPlasmaC v*N.XXX.YYY* - powered by QtVCP on LinuxCNC vZ.Z.Z", wobei *N* die Version von QtPlasmaC, *XXX* die Version der HAL-Komponente (PlasmaC.comp), *YYY* die GUI-Version und *Z.Z.Z* die Version von LinuxCNC ist.

Einheitensysteme

Alle Einstellungen und Parameter in QtPlasmaC müssen in den gleichen Einheiten wie in der <*Maschinenname*>.ini-Datei angegeben sein, also entweder metrisch oder imperial.

Wenn der Benutzer versucht, eine G-Code-Datei auszuführen, die sich im „anderen“ Einheitensystem befindet, müssen alle Parameter, einschließlich der Materialdateiparameter, immer noch in den nativen Maschineneinheiten vorliegen. Alle weiteren Konvertierungen, die zum Ausführen der G-Code-Datei erforderlich sind, werden automatisch vom G-Code-Filterprogramm durchgeführt.

Beispiel: Wenn ein Benutzer eine metrische Maschine hat und eine G-Code-Datei ausführen möchte, die für das Schneiden von 1/4" dickem Material in zölligen Einheiten (Zoll - G20) eingerichtet ist, dann muss der Benutzer mit der metrischen Maschine sicherstellen, dass entweder die Materialnummer in der G-Code-Datei auf das entsprechende zu schneidende metrische Material eingestellt ist oder dass ein neues Material mit den richtigen metrischen Parametern für das zu schneidende metrische Material erstellt wird. Wenn der metrische Benutzer die G-Code-Datei mit zölligem Material schneiden wollte, müssten die neuen Materialparameter bei ihrer Eingabe von zölligen Einheiten in metrische umgewandelt werden.

Präambel und Postambel Codes

Die folgenden Strophen sind das Minimum an empfohlenen Codes, die in der Präambel und Postambel jeder G-Code-Datei enthalten sein sollten, die von QtPlasmaC ausgeführt werden soll:

Metrisch:

```
G21 G40 G49 G64p0.1 G80 G90 G92.1 G94 G97
```

Imperial:

```
G20 G40 G49 G64p0.004 G80 G90 G92.1 G94 G97
```

Eine ausführliche Erläuterung der einzelnen G-Codes finden Sie unter dem Link docs:../gcode/g-code.html[hier].

Beachten Sie, dass in diesem Benutzerhandbuch mehrere zusätzliche Empfehlungen für Codes gegeben werden, die je nach den vom Benutzer gewünschten Funktionen sowohl in der Präambel als auch in der Postambel hinzugefügt werden sollten.

Obligatorische Codes

Abgesehen vom Präambelcode, Postambelcode und X/Y-Bewegungscode ist die einzige obligatorische G-Code-Syntax für QtPlasmaC zur Ausführung eines G-Code-Programms mit einem Brenner zum

Schneiden `M3 $0 S1`, um einen Schnitt zu beginnen, und `M5 $0`, um einen Schnitt zu beenden.

Aus Gründen der Abwärtskompatibilität ist es zulässig, `M3 S1` anstelle von `M3 $0 S1` zu verwenden, um einen Schneidauftrag zu beginnen, und `M5` anstelle von `M5 $0`, um einen Schneidauftrag zu beenden. Beachten Sie, dass dies nur für Schneidaufträge gilt, für Ritz- und Tuschieraufträge ist die Werkzeugkennung `$n` obligatorisch.

Koordinaten

Siehe empfohlene Z-Achse Einstellungen.

Jedes Mal, wenn LinuxCNC (QtPlasmaC) gestartet wird, ist eine gemeinsame Referenzfahrt erforderlich. Dies ermöglicht LinuxCNC (QtPlasmaC), um die bekannten Koordinaten der einzelnen Achsen zu etablieren und die weichen Grenzen auf die Werte in der <*machinenname*>.ini Datei angegeben, um die Maschine von Absturz in einen harten Anschlag während des normalen Gebrauchs zu verhindern.

Wenn die Maschine keine Referenzfahrtschalter hat, muss der Benutzer sicherstellen, dass sich alle Achsen vor der Referenzfahrt an den in der Datei <*maschinenname*>.ini angegebenen Referenzpunktkoordinaten befinden.

Wenn die Maschine mit Referenzfahrtschaltern ausgestattet ist, fährt sie zu den angegebenen Referenzpunktkoordinaten, wenn die Gelenke referenziert werden.

Je nach Konfiguration der Maschine gibt es entweder eine Schaltfläche **Home All** oder jede Achse muss einzeln referenziert werden. Benutzen Sie die entsprechende(n) Taste(n), um die Maschine zu referenzieren.

Wie im Abschnitt zur Ersteinrichtung erwähnt, wird empfohlen, dass der Benutzer bei der ersten Verwendung von QtPlasmaC sicherstellt, dass sich unter der Taschenlampe nichts befindet, und dann die Z-Achse nach unten zieht, bis sie an der Z-Achse anhält, MINIMUM_LIMIT dann auf die 0 neben der Z-Achsen-DRO klicken, um **Touch Off** mit der Z-Achse auszuwählen, um die Z-Achse auf Null zu setzen. Dies sollte nicht erneut getan werden müssen.

Wenn der Benutzer beabsichtigt, das Material jedes Mal an der exakt gleichen Stelle auf dem Tisch zu platzieren, könnte er die X- und Y-Achsen der Maschine auf die entsprechende X0-Y0-Position verfahren, wie sie von der CAM-Software festgelegt wurde, und dann beide Achsen mit einem Null-Offset **abtasten**.

Wenn der Benutzer beabsichtigt, das Material willkürlich auf dem Tisch zu platzieren, muss er die X- und Y-Achse an der entsprechenden Position **abtasten**, bevor er das Programm startet.

Schnitt Vorschubgeschwindigkeit

QtPlasmaC ist in der Lage, eine Materialdatei zu lesen, um alle erforderlichen Schnittparameter zu laden. Damit die G-Code-Datei die Schnittvorschubeinstellung aus den Schnittparametern verwenden kann, verwenden Sie den folgenden Code in der G-Code-Datei:

```
F#<_hal[plasmac.cut-feed-rate]>
```

Es ist möglich, den Standard-G-Code **F** zu verwenden, um die Schnittvorschubgeschwindigkeit wie folgt

einzustellen:

```
F 1000
```

Wenn das **F**-Wort verwendet wird und der Wert des **F**-Wortes nicht mit dem Schnittvorschub des ausgewählten Materials übereinstimmt, wird beim Laden der G-Code-Datei ein Warndialog angezeigt.

Material-Datei

Die Materialverwaltung verwendet eine Materialdatei, die für die Maschinenkonfiguration erstellt wurde, als der Konfigurationsassistent ausgeführt wurde, und ermöglicht es dem Benutzer, bekannte Materialeinstellungen bequem zu speichern, um sie entweder manuell oder automatisch über G-Code abzurufen. Die resultierende Material-Datei trägt den Namen <***Maschinen_name***>**_material.cfg**.

QtPlasmaC erfordert nicht die Verwendung einer Materialdatei. Stattdessen kann der Benutzer die Schnittparameter manuell im Abschnitt MATERIAL der PARAMETER-Registerkarte ändern. Es ist auch nicht erforderlich, die automatischen Materialänderungen zu verwenden. Wenn der Benutzer diese Funktion nicht nutzen möchte, kann er die Materialänderungscodes in der G-Code-Datei einfach weglassen.

Es ist auch möglich, die Materialdatei nicht zu verwenden und material automatisch laden aus der G-Code-Datei zu verwenden.

Die Materialnummern in der Materialdatei müssen nicht fortlaufend sein und auch nicht in numerischer Reihenfolge stehen.

Die folgenden Variablen sind obligatorisch und eine Fehlermeldung wird angezeigt, wenn sie beim Laden der Materialdatei nicht gefunden werden.

- PIERCE_HEIGHT
- PIERCE_DELAY
- CUT_HEIGHT
- CUT_SPEED

NOTE	Wenn Rohrschneiden mit dem magischen Kommentar #<tube_cut>=1 durchgeführt wird, ist nur die Variable PIERCE_DELAY zwingend erforderlich, alle anderen Variablen sind optional.

Die folgenden Variablen sind optional. Werden sie nicht erkannt oder ist ihnen kein Wert zugewiesen, wird ihnen der Wert 0 zugewiesen und es erscheint keine Fehlermeldung.

- NAME
- KERF_WIDTH
- THC
- PUDDLE_JUMP_HEIGHT
- PUDDLE_JUMP_DELAY

- CUT_AMPS
- CUT_VOLTS
- PAUSE_AT_END
- GAS_PRESSURE
- CUT_MODE

NOTE Die Materialnummern 1000000 und höher sind für temporäre Materialien reserviert.

WARNING Es liegt in der Verantwortung des Anwenders, dafür zu sorgen, dass die Variablen einbezogen werden, wenn sie eine Voraussetzung für die Ausführung des G-Codes sind.

Die Materialdatei hat das folgende Format:

```
[MATERIAL_NUMBER_1]
NAME                = name
KERF_WIDTH          = value
THC                 = value (0 = off, 1 = on)
PIERCE_HEIGHT       = value
PIERCE_DELAY        = value
PUDDLE_JUMP_HEIGHT  = value
PUDDLE_JUMP_DELAY   = value
CUT_HEIGHT          = value
CUT_SPEED           = value
CUT_AMPS            = value (for info only unless PowerMax communications is enabled)
CUT_VOLTS           = value (modes 0 & 1 only, if not using auto voltage sampling)
PAUSE_AT_END        = value
GAS_PRESSURE        = value (only used for PowerMax communications)
CUT_MODE            = value (only used for PowerMax communications)
```

Es ist möglich, neues Material hinzuzufügen, Material zu löschen oder vorhandenes Material auf der Registerkarte mit PARAMETERN zu bearbeiten. Es ist auch möglich, dies durch die Verwendung von magic comments in einer G-Code-Datei zu erreichen.

Die Materialdatei kann mit einem Texteditor bearbeitet werden, während LinuxCNC läuft. Nachdem alle Änderungen gespeichert wurden, drücken Sie **Reload** im Abschnitt MATERIAL der PARAMETER-Registerkarte, um die Materialdatei neu zu laden.

Manuelles Materialhandling

Bei der manuellen Materialhandhabung würde der Benutzer das Material manuell aus der Materialliste im Abschnitt MATERIAL der PARAMETER-Registerkarte auswählen, bevor er das G-Code-Programm startet. Zusätzlich zur Auswahl von Materialien aus der Materialliste im Abschnitt MATERIAL der PARAMETER-Registerkarte kann der Benutzer die MDI verwenden, um Materialien mit dem folgenden Befehl zu ändern:

```
M190 Pn
```

Der folgende Code ist der Mindestcode, der für einen erfolgreichen Schnitt mit der manuellen Materialauswahlmethode erforderlich ist:

```
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

NOTE Bei der manuellen Materialhandhabung kann der Benutzer nur ein Material für die gesamte Arbeit verwenden.

Automatisches Materialhandling

Für die automatische Materialhandhabung fügt der Benutzer seiner G-Code-Datei Befehle hinzu, die es QtPlasmaC ermöglichen, das Material automatisch zu ändern.

Die folgenden Codes können verwendet werden, um QtPlasmaC einen automatischen Materialwechsel zu ermöglichen:

- **M190 P*n*** - Ändert das aktuell angezeigte Material auf die Materialnummer *n*.
- **M66 P3 L3 Q1** - Fügt eine kleine Verzögerung hinzu (1 Sekunde in diesem Beispiel), um zu warten, bis QtPlasmaC bestätigt, dass es erfolgreich Materialien gewechselt hat.
- **F#<_hal[plasmac.cut-feed-rate]>** - Setzt den Schnittvorschub auf den Vorschub, der im Abschnitt MATERIAL der PARAMETER-Registerkarte angegeben ist.

Für die automatische Materialverarbeitung MÜSSEN die Codes in der angegebenen Reihenfolge angewendet werden. Wenn ein G-Code-Programm geladen wird, das einen oder mehrere Materialwechselbefehle enthält, wird das erste Material in der oberen Kopfzeile des VORSCHAU-FENSTERS auf der Haupt-Registerkarte (engl. main) angezeigt, während das Programm geladen wird.

Minimaler Code, der für einen erfolgreichen Schnitt mit der automatischen Materialauswahlmethode erforderlich ist:

```
M190 Pn
M66 P3 L3 Q1
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

NOTE Die Rückkehr zum Standardmaterial vor Ende des Programms ist mit dem Code **M190 P-1** möglich.

Material hinzufügen Via Magic Kommentare in G-Code

Durch die Verwendung von "magischen Kommentaren" in einer G-Code-Datei ist es möglich, das

Folgende zu tun:

- Fügen Sie der Datei <*Maschinenname*>_material.cfg neue Materialien hinzu.
- Bearbeiten Sie vorhandene Materialien in der Datei <*Maschinenname*>_material.cfg.
- Verwendung eines oder mehrerer vorübergehender Materialien.

Temporäre Materialien werden von QtPlasmaC automatisch nummeriert und der Materialwechsel wird ebenfalls von QtPlasmaC durchgeführt und sollte nicht durch CAM-Software oder anderweitig zur G-Code-Datei hinzugefügt werden. Die Materialnummern beginnen bei 1000000 und werden für jedes temporäre Material hochgezählt. Es ist nicht möglich, ein temporäres Material zu speichern. Der Benutzer kann jedoch ein neues Material erstellen, während ein temporäres Material angezeigt wird, und es wird die Einstellungen des temporären Materials als Standardwerte verwenden.

TIP Es ist möglich, nur temporäre Materialien zu verwenden und eine leere <*Maschinenname*>_material.cfg-Datei zu haben. Dadurch entfällt die Notwendigkeit, die QtPlasmaC-Materialdatei mit der CAM-Werkzeugdatei zu aktualisieren.

- Der gesamte Kommentar muss in Klammern gesetzt werden.
- Der Anfang des magischen Kommentars muss lauten: **(o=**
- Das Gleichheitszeichen muss unmittelbar nach jedem Parameter stehen, ohne Leerzeichen.
- Die obligatorischen Parameter müssen im magischen Kommentar enthalten sein (für Option 0 ist **na** optional und **nu** wird nicht verwendet).
- Eine G-Code-Datei kann eine beliebige Anzahl und Art von magischen Kommentaren enthalten.
- Wenn die Option 0 zusätzlich zu Option 1 und/oder Option 2 verwendet werden soll, müssen alle Optionen 0 nach allen Optionen 1 oder allen Optionen 2 in der G-Code-Datei erscheinen.

Die Optionen sind:

Option	Beschreibung
0	Erzeugt ein temporäres Standardmaterial. Die mit dieser Option hinzugefügten Materialinformationen werden durch einen LinuxCNC-Neustart oder ein Neuladen der Materialien verworfen. Sie können auch durch eine neue G-Code-Datei überschrieben werden, die temporäre Materialien enthält.
1	Fügt ein neues Material hinzu, wenn die angegebene Nummer nicht vorhanden ist.
2	Überschreibt ein vorhandenes Material, wenn die angegebene Nummer existiert. Fügt ein neues Material hinzu, wenn die angegebene Nummer nicht vorhanden ist.

Obligatorische Parameter sind:

Name	Beschreibung
o	Wählt die zu verwendende Option aus.
nu	Legt die Materialnummer fest (wird bei Option 0 nicht verwendet).
na	Legt den Materialnamen fest (optional für Option 0).
ph	Legt die Höhe des Durchstichs fest.
pd	Legt die Durchdringungsverzögerung fest.
ch	Legt die Schnitthöhe fest.
fr	Legt die Vorschubgeschwindigkeit fest.

Optionale Parameter sind:

Name	Beschreibung
mt	Bestimmt die Materialstärke (engl. material thickness).
kw	Legt die Schnittspaltbreite (engl. kerf width) fest.
th	Legt den THC-Status fest (0=deaktiviert, 1=aktiviert).
ca	Legt die Stromstärke für das Schneiden fest (engl. cut amps) fest.
cv	Setzt die Spannung für das Schneiden (engl. cut voltage).
pe	Legt die Verzögerung für die Pause am Ende fest.
gp	Stellt den Gasdruck ein (PowerMax).
cm	Legt den Schneid-Modus (engl. cut mode) fest (PowerMax).
jh	Legt die Pfützensprunghöhe fest.
jd	Legt die Pfützensprungverzögerung fest.

Ein vollständiges Beispiel (metrisch):

```
(o=0, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, mt=5, kw=0.5, th=1,
ca=40, cv=110, pe=0.1, gp=5, cm=1, jh=0, jd=0)
```

Ein vollständiges Beispiel (Zoll):

```
(o=0, nu=2, na=0.197" Mild Steel 40A, ph=0.122, pd=0.1, ch=0.029, fr=118, mt=0.197,
kw=0.020, th=1, ca=40, cv=110, pe=0.1, gp=72, cm=1, jh=0, jd=0)
```

Wenn in einer G-Code-Datei ein temporäres Material angegeben wurde, werden die Zeilen für Materialwechsel (M190...) und Warten auf Wechsel (M66...) vom G-Code-Filter hinzugefügt und sind in der G-Code-Datei nicht erforderlich.

Material Konverter

Diese Anwendung dient der Konvertierung bestehender Werkzeugtabellen in QtPlasmaC Materialdateien. Sie kann auch eine Materialdatei aus manuellen Benutzereingaben in Eingabefeldern erstellen.

In diesem Stadium sind nur Konvertierungen für Werkzeugtabellen verfügbar, die aus SheetCam oder Fusion 360 exportiert wurden.

SheetCam-Werkzeugtabellen sind vollständig und die Konvertierung erfolgt vollautomatisch. Die SheetCam-Werkzeugdatei muss im SheetCam .tools-Format vorliegen.

Fusion 360-Werkzeugtabellen verfügen nicht über alle erforderlichen Felder, weshalb der Benutzer zur Eingabe fehlender Parameter aufgefordert wird. Die Fusion 360-Werkzeugdatei muss im JSON-Format von Fusion 360 vorliegen.

Wenn der Benutzer ein Format aus einer anderen CAM-Software hat, das er konvertiert haben möchte, erstellen Sie ein **Neues Thema** im Abschnitt PlasmaC-Forum des LinuxCNC-Forum, um diese Ergänzung zu beantragen.

Der Materialkonverter kann mit einer der beiden folgenden Methoden von einem Terminal aus gestartet werden.

Geben Sie für eine Paketinstallation (Buildbot) den folgenden Befehl in einem Terminalfenster ein:

```
qtplasmac-materials
```

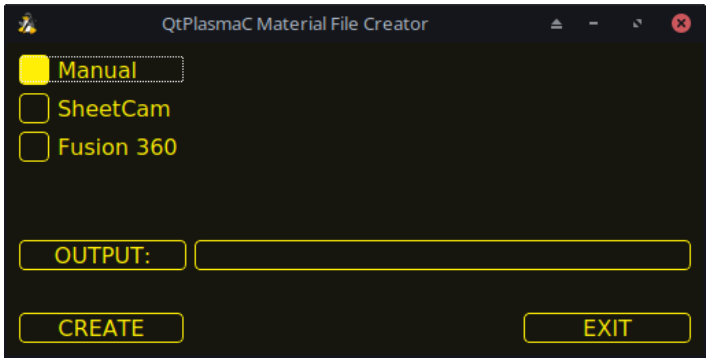
Geben Sie für eine "run in place"-Installation die folgenden beiden Befehle in ein Terminalfenster ein:

```
source ~/linuxcnc-dev/scripts/rip-environment
qtplasmac-materials
```

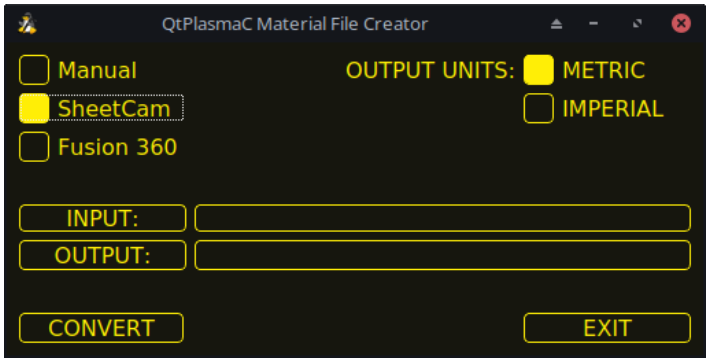
Daraufhin wird das Hauptdialogfeld Materialkonverter mit der Standardeinstellung Manuell angezeigt.

Wählen Sie eines aus:

- **Manuell** (engl. manual) - um manuell eine neue Materialdatei zu erstellen.

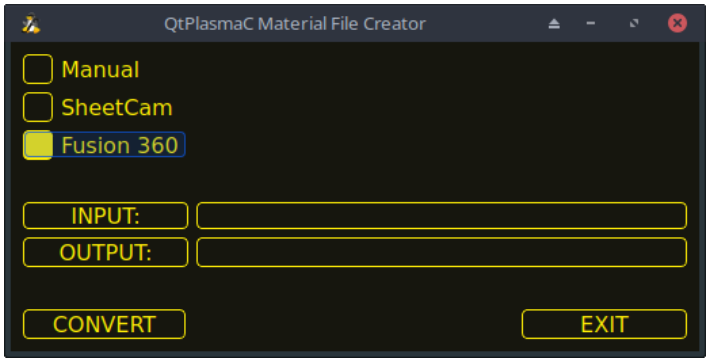


- **SheetCam** - um eine SheetCam-Werkzeugdatei zu konvertieren.



Wählen Sie nur für SheetCam, ob der Benutzer eine metrische oder imperiale Ausgabedatei benötigt.

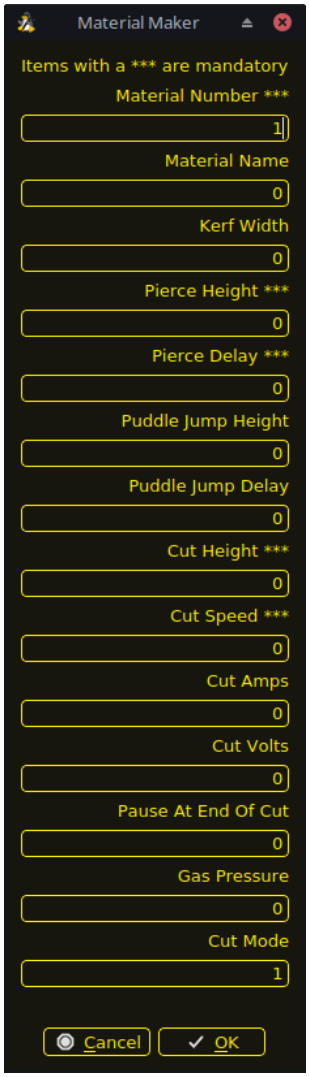
- **Fusion 360** - zum Konvertieren einer Fusion 360 Werkzeugdatei.



Um zu konvertieren:

1. Wählen Sie die zu konvertierende Eingabedatei, drücken Sie **INPUT**, um eine Dateiauswahl aufzurufen, oder geben Sie die Datei direkt in das Eingabefeld ein.
2. Wählen Sie die Ausgabedatei, in die geschrieben werden soll, drücken Sie **OUTPUT**, um eine Dateiauswahl aufzurufen, oder geben Sie die Datei direkt in das Eingabefeld ein. Normalerweise ist dies ~/linuxcnc/configs/<*Maschinenname*>_material.cfg. Falls erforderlich, kann der Benutzer eine andere Datei auswählen und die Datei <*Maschinenname*>_material.cfg manuell bearbeiten.
3. Klicken Sie auf **CREATE/CONVERT** und die neue Materialdatei wird erstellt.

Sowohl bei einer manuellen Erstellung als auch bei einer Fusion 360-Konvertierung wird ein Dialogfeld mit allen verfügbaren Parametern angezeigt, die eingegeben werden können. Jeder mit *** markierte Eintrag ist obligatorisch, alle anderen Einträge sind je nach den Konfigurationsanforderungen des Benutzers optional.



NOTE Wenn der Benutzer ~/linuxcnc/configs/<*Maschinenname*>_material.cfg auswählt und die Datei bereits existiert, wird sie überschrieben.

LASER

QtPlasmaC hat die Möglichkeit, einen Laser zum Setzen des Ursprungs mit oder ohne Rotationskompensation zu verwenden. Mit einer Rotationskompensation kann der Arbeitsversatz auf ein Materialblatt mit nicht parallel zu den X/Y-Achsen der Maschine verlaufenden Kanten ausgerichtet werden. Die LASER-Schaltfläche wird aktiviert, nachdem die Maschine referenziert wurde. Dieser Button wird nicht sichtbar bevor ein LASER Offset in der <*Maschinenname*>.prefs Datei festgelegt wurde.

Um diese Funktion zu nutzen, muss der Benutzer den Versatz des Lasers von der Brennermitte einstellen, indem er das unter Peripherie-Offsets beschriebene Verfahren befolgt.

Um die Offsets manuell zu ändern, kann der Benutzer eine oder beide der folgenden Optionen im Abschnitt **[LASER_OFFSET]** der Datei <*machine_name*>.prefs bearbeiten:

```
X axis = n.n
Y axis = n.n
```

wobei *n.n* der Abstand zwischen der Mittellinie des Brenners und dem Fadenkreuz des Lasers ist.

Zusätzlich kann der Laser über einen HAL-Pin mit folgendem Namen an einen beliebigen Ausgang angeschlossen werden, um den Laser ein- und auszuschalten:

```
qtplasmac.laser_on
```

Ursprung mit Null-Drehung festlegen:

1. Klicken Sie auf die Schaltfläche **LASER**.
2. Die Beschriftung des Buttons **LASER** ändert sich zu **MARK EDGE** und der HAL-Pin namens qtplasmac.laser_on wird eingeschaltet.
3. Bewegen Sie sich, bis sich das Fadenkreuz des Lasers über dem gewünschten Ursprungspunkt befindet.
4. Drücken Sie **MARK EDGE**. Die Beschriftung der Taste **MARK EDGE** ändert sich in **SET ORIGIN**.
5. Drücken Sie **SET ORIGIN** (engl. für Ursprung festlegen). Die Beschriftung der Taste **SET ORIGIN** ändert sich in **MARK EDGE** und der HAL-Pin mit dem Namen qtplasmac.laser_on wird deaktiviert.
6. Der Brenner fährt nun in die Position X0 Y0.
7. Der Offset ist nun erfolgreich gesetzt.

Ursprung mit Drehung festlegen:

1. Klicken Sie auf die Schaltfläche **LASER**.
2. Die Beschriftung des Buttons **LASER** ändert sich zu **MARK EDGE** und der HAL-Pin namens qtplasmac.laser_on wird eingeschaltet.
3. Rütteln Sie so lange, bis sich das Fadenkreuz des Lasers an der Kante des Materials in einem angemessenen Abstand zum gewünschten Ursprungspunkt befindet.
4. Drücken Sie **MARK EDGE**. Die Beschriftung der Taste **MARK EDGE** ändert sich in **SET ORIGIN**.
5. Bewegen Sie sich, bis sich das Fadenkreuz des Lasers am Ursprungspunkt des Materials befindet.
6. Drücken Sie **SET ORIGIN** (engl. für Ursprung festlegen). Die Beschriftung der Taste **SET ORIGIN** ändert sich in **MARK EDGE** und der HAL-Pin mit dem Namen qtplasmac.laser_on wird deaktiviert.
7. Der Brenner fährt nun in die Position X0 Y0.
8. Der Offset ist nun erfolgreich eingestellt.

Um den Laser auszuschalten und eine Ausrichtung abzubrechen:

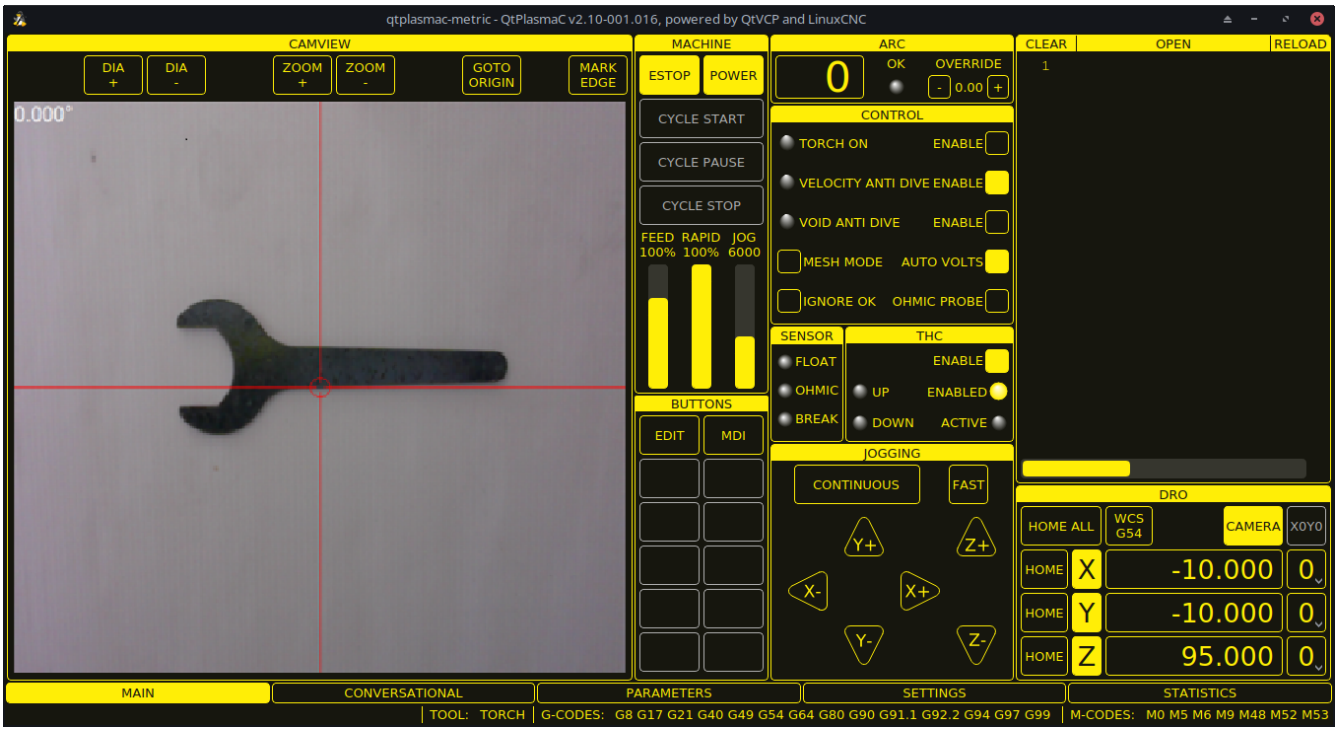
1. Drücken Sie die Taste **LASER** und halten Sie sie länger als 750 ms. gedrückt.
2. Die Beschriftung der Schaltfläche **LASER** ändert sich in **LASER** und der HAL-Pin mit dem Namen qtplasmac.laser_on wird ausgeschaltet.
3. Lassen Sie die Taste **LASER** los.

Wenn ein Ausrichtungslaser eingerichtet wurde, ist es möglich, den Laser während CUT RECOVERY zur genauen Positionierung der neuen Startkoordinaten zu verwenden.

Für einen Probelauf (engl. dry run) des G-codes mit dem Laser: . Stellen Sie sicher, dass es keine

Begrenzungs-Fehler (engl. bounds errors) gibt und ZYKLUS START aktiviert ist. . Drücken Sie die Taste **LASER** und halten Sie sie länger als 750 ms gedrückt, der Laser wird eingeschaltet und der Trockendurchlauf (engl. dry run) wird gestartet. . Lassen Sie die Taste **LASER** los.

KAMERA



QtPlasmaC hat die Möglichkeit, eine USB-Kamera zu verwenden, um den Ursprung mit oder ohne Rotationskompensation festzulegen. Mit einer Rotationskompensation kann der Arbeitsversatz auf ein Materialblatt mit nicht parallel zu den X/Y-Achsen der Maschine verlaufenden Kanten ausgerichtet werden. Der CAMERA-Button wird aktiviert, nachdem das Gerät referenziert wurde. Dieser Button wird nicht sichtbar bevor ein CAMERA Offset in der <*Maschinenname*>.prefs Datei festgelegt wurde.

Um diese Funktion zu nutzen, muss der Benutzer den Versatz der Kamera von der Brennermitte einstellen, indem er das unter Peripherie-Offsets beschriebene Verfahren befolgt.

Um die Offsets manuell zu ändern, kann der Benutzer eine oder beide der folgenden Achsen-Optionen im Abschnitt **[CAMERA_OFFSET]** der Datei <*machine_name*>.prefs bearbeiten:

```
X axis = n.n
Y axis = n.n
Camera port = 0
```

wobei *n.n* der Abstand von der Mittellinie des Brenners zum Fadenkreuz der Kamera ist.

Ursprung mit Null-Drehung festlegen:

1. Bewegen Sie die Maus, bis sich das Fadenkreuz über dem gewünschten Ursprungspunkt befindet.
2. Drücken Sie **MARK EDGE**. Die Beschriftung der Schaltfläche **MARK EDGE** ändert sich in **SET ORIGIN** und die Schaltfläche **GOTO ORIGIN** wird deaktiviert.

3. Drücken Sie **Ursprung festlegen**. Die Beschriftung der Schaltfläche **Herkunft festlegen** ändert sich in **KANTE MARKIEREN** und die Schaltfläche **Herkunft gehen** wird aktiviert.
4. Der Brenner fährt nun in die Position X0 Y0.
5. Der Offset ist nun erfolgreich gesetzt.

Ursprung mit Drehung festlegen:

1. Bewegen Sie das Fadenkreuz, bis es sich am Rand des Materials in einem angemessenen Abstand zum gewünschten Ursprungspunkt befindet.
2. Drücken Sie **MARK EDGE**. Die Beschriftung der Schaltfläche **MARK EDGE** ändert sich in **SET ORIGIN** und die Schaltfläche **GOTO ORIGIN** wird deaktiviert.
3. Bewegen Sie die Maus, bis sich das Fadenkreuz am Ausgangspunkt des Materials befindet.
4. Drücken Sie **Ursprung festlegen**. Die Beschriftung der Schaltfläche **Herkunft festlegen** ändert sich in **KANTE MARKIEREN** und die Schaltfläche **Herkunft gehen** wird aktiviert.
5. Der Brenner fährt nun in die Position X0 Y0.
6. Der Offset ist nun erfolgreich eingestellt.

Im CAMVIEW-Bedienfeld kann die Maus das Fadenkreuz und die Zoomstufe wie folgt beeinflussen:

- Mausrad scrollen - Durchmesser des Fadenkreuzes ändern.
- Doppelklick mit der Mausradtaste - Stellt den Fadenkreuzdurchmesser auf den Standardwert zurück.
- Linke Maustaste gedrückt + Scrollrad - Ändert die Zoomstufe der Kamera.
- Klicken mit der linken Maustaste + Doppelklick mit der Radtaste - Stellt die Standard-Zoomstufe der Kamera wieder her.

Pfadtoleranz

Die Pfad-/Bahntoleranz wird mit einem G64-Befehl und einem nachfolgenden P-Wert eingestellt. Der P-Wert entspricht dem Betrag, um den die tatsächliche Schnittbahn, der die Maschine folgt, von der programmierten Schnittbahn abweichen darf.

Die Standard-LinuxCNC Bahntoleranz ist für die maximale Geschwindigkeit, die stark runden Ecken, wenn sie mit normalen Plasma-Schneidgeschwindigkeiten verwendet wird eingestellt.

Es wird empfohlen, die Pfad-Toleranz durch Einfügen des entsprechenden G64-Befehls und des P-Werts in den Kopf jeder G-Code-Datei festzulegen.

Das bereitgestellte G-Code-Filterprogramm wird vor dem ersten Bewegungsbefehl auf die Existenz eines `G64 P_n__`-Befehls testen. Wenn kein G64-Befehl gefunden wird, so wird ein `G64 P0.1`-Befehl eingefügt, der die Wegtoleranz auf 0,1 mm setzt. Für eine imperiale Konfiguration wird der Befehl `G64 P0.004` sein.

Für metrisch:

```
G64 P0.1
```

Für imperial:

```
G64 P0.004
```

Angehaltene Bewegung

QtPlasmaC ermöglicht die Neupositionierung der X- und Y-Achse entlang des aktuellen Schnittpfades, während das G-Code-Programm pausiert, unterstützend bei der Cut Recovery.

Um diese Funktion nutzen zu können, muss die adaptive Vorschubsteuerung (M52) von LinuxCNC eingeschaltet sein (P1). Dies ist auch eine Voraussetzung für die Reduktion der Geschwindigkeit beim Loch-Schneiden.

Um **Paused Motion** zu aktivieren, muss die Präambel des G-Codes die folgende Zeile enthalten:

```
M52 P1
```

Um **Paused Motion** an einem beliebigen Punkt zu deaktivieren, verwenden Sie den folgenden Befehl:

```
M52 P0
```

Pause am Ende von Cut

Diese Funktion kann verwendet werden, damit der Lichtbogen die Brennerposition "einholen" kann, um den Schnitt vollständig zu beenden. Sie ist in der Regel für dickere Materialien erforderlich und ist besonders nützlich beim Schneiden von Edelstahl.

Die Verwendung dieser Funktion bewirkt, dass alle Bewegungen am Ende des Schnitts angehalten werden, während der Brenner noch eingeschaltet ist. Nach Ablauf der Verweilzeit (in Sekunden), die mit dem Parameter **Pause At End** im Abschnitt MATERIAL der PARAMETER-Registerkarte eingestellt wurde, fährt QtPlasmaC mit dem Befehl M5 fort, um den Brenner auszuschalten und anzuheben.

Mehrere Werkzeuge

QtPlasmaC hat die Fähigkeit, die Verwendung von mehr als einer Art von Plasma-Werkzeug durch die Verwendung von LinuxCNC Spindeln als Plasma-Werkzeug bei der Ausführung eines G-Code-Programm zu ermöglichen.

Gültige Plasmageräte für den Einsatz sind:

Name	Werkzeug #	Beschreibung
Plasma-Brenner	0	Wird für normales Plasmaschneiden verwendet.

Name	Werkzeug #	Beschreibung
Schreiber (engl. scribe)	1	Wird für die Materialgravur verwendet.
Plasma-Brenner	2	Wird zum Tupfen (Erzeugen von Vertiefungen zur Unterstützung des Bohrens) verwendet.

Eine LinuxCNC Spindelnummer (bezeichnet durch $*n*) ist erforderlich, um in den Start-Befehl und auch die Ende-Befehl, um in der Lage zu starten und stoppen Sie die richtige Plasma-Werkzeug sein. Beispiele:

- Mit `M3 $0 S1` wird das Plasmaschneidwerkzeug ausgewählt und gestartet.
- Mit `M3 $1 S1` wird der Schreiber (engl. scribe) ausgewählt und gestartet.
- Mit `M3 $2 S1` wird das Plasmapunktiergerät ausgewählt und gestartet.
- Mit `M5 $0` wird das Plasmaschneidwerkzeug angehalten.
- Mit `M5 $1` wird der Schreiber gestoppt.
- Mit `M5 $2` wird das Plasmapunktiergerät gestoppt.

Es ist zulässig, **M5 $-1** anstelle der obigen M5 $*n*-Codes zu verwenden, um alle Werkzeuge anzuhalten.

Um einen Ritz zu verwenden, ist es notwendig für den Benutzer, um die X-und Y-Achse Offsets auf die LinuxCNC Werkzeugtabelle hinzuzufügen. Werkzeug 0 ist dem Plasmabrenner und Werkzeug 1 ist dem Ritzer zugewiesen. Werkzeuge werden mit einem **T*n* M6** Befehl ausgewählt, und dann ein **G43 H0** Befehl ist erforderlich, um die Offsets für das ausgewählte Werkzeug anzuwenden. Es ist wichtig zu beachten, dass die LinuxCNC-Werkzeugtabelle und die Werkzeugbefehle nur dann ins Spiel kommen, wenn der Benutzer zusätzlich zu einem Plasmabrenner einen scribe verwendet. Für weitere Informationen, siehe scribe.

Geschwindigkeitsreduzierung

Es gibt einen HAL-Pin mit der Bezeichnung **motion.analog-out-03**, der im G-Code mit den Befehlen **M67 (Synchronisiert mit Bewegung)/M68 (Sofort)** geändert werden kann. Mit diesem Pin wird die Geschwindigkeit des ursprünglichen Vorschubs auf den im Befehl angegebenen Prozentsatz reduziert.

Die „VEL:“-Anzeige oben rechts im Vorschaufenster wird aktualisiert, um den Prozentsatz der ursprünglich programmierten Vorschubgeschwindigkeit anzugeben. Beispiel: „VEL@20%:“ bedeutet, dass der Tisch mit 20 % der programmierten Vorschubgeschwindigkeit schneidet – also einer Reduktion um 80 %.

NOTE Wegen der unterschiedlichen Auffrischungs-Rate (engl. polling intervals) zwischen der GUI und der PlasmaC-Komponente ist das Update des die Geschwindigkeit anzeigenden Labels zeitlich möglicherweise versetzt (typischerweise bis zu 100 ms später).

Es ist wichtig, den Unterschied zwischen **Synchronisiert mit Bewegung** und **Sofort** gründlich zu verstehen:

- `M67` (Synchronisiert mit Bewegung) - Die tatsächliche Änderung des angegebenen Ausgangs (z. B. P2

(THC)) erfolgt zu Beginn des nächsten Bewegungsbefehls. Wenn es keinen nachfolgenden Bewegungsbefehl gibt, werden die Ausgangsänderungen nicht durchgeführt. Es empfiehlt sich, einen Bewegungscode (z. B. G0 oder G1) direkt nach einem M67 zu programmieren.

- `M68` (Immediate) - Diese Befehle werden sofort ausgeführt, wenn sie vom Motion Controller empfangen werden. Da sie nicht mit der Bewegung synchronisiert sind, unterbrechen sie das Blending. Das heißt, wenn diese Codes inmitten von aktiven Bewegungscodes verwendet werden, wird die Bewegung angehalten, um diese Befehle zu aktivieren.

Beispiele:

- Mit `M67 E3 Q0` würde die Geschwindigkeit auf 100% der **CutFeedRate** gesetzt.
- Mit `M67 E3 Q40` würde die Geschwindigkeit auf 40% der **CutFeedRate** gesetzt.
- Mit `M67 E3 Q60` würde die Geschwindigkeit auf 60% der **CutFeedRate** gesetzt.
- `M67 E3 Q100` würde die Geschwindigkeit auf 100% von **CutFeedRate** setzen.

Q Werte die gleiner oder gleich 0 sind oder größer als 100 werden auf 100 gesetzt.

Wenn der Benutzer diese Funktion nutzen möchte, wäre es ratsam, `M68 E3 Q0` sowohl in die Präambel als auch in die Postambel des G-Code-Programms einzufügen, damit die Maschine in einem bekannten Zustand startet und endet.

IMPORTANT	**G-CODE THC** UND **VELOCITY BASED THC** KÖNNEN NICHT VERWENDET WERDEN, WENN **CUTTER COMPENSATION** IN KRAFT IST; ES WIRD EINE FEHLERMELDUNG ANGEZEIGT.
WARNING	Wenn Cut Feed Rate im Abschnitt MATERIAL der PARAMETER-Registerkarte auf Null gesetzt ist, dann verwendet QtPlasmaC **motion.requested-velocity** (wie durch einen Standard Vorschub-Aufruf im G-Code eingestellt) für die THC-Berechnungen. Dies wird nicht empfohlen, da es kein zuverlässiger Weg ist, geschwindigkeitsbasierte THC zu implementieren.
NOTE	Alle Verweise auf CutFeedRate beziehen sich auf den Wert **Cut Feed Rate**, der im Abschnitt MATERIAL der PARAMETER-Registerkarte angezeigt wird.

THC (Brennerhöhensteuerung, engl. torch height controller)

Die THC kann über den THC-Rahmen des Haupt-Registerkarte (engl. main tab) aktiviert oder deaktiviert werden.

Die THC kann auch direkt über das G-Code-Programm aktiviert oder deaktiviert werden.

Die THC wird erst dann aktiv, wenn die Geschwindigkeit 99,9 % der **CutFeedRate** erreicht hat und die THC **Delay**-Zeit (falls vorhanden) im THC-Abschnitt der PARAMETER-Registerkarte abgelaufen ist. Dies dient dazu, dass sich die Lichtbogenspannung stabilisieren kann.

QtPlasmaC verwendet eine Steuerspannung, die vom Zustand der Checkbox **AUTO VOLTS** auf dem

Haupt-Registerkarte (engl. MAIN Tab) abhängig ist:

1. Wenn **Use Auto Volts** aktiviert ist, wird die tatsächliche Abschaltspannung am Ende der THC **Delay** -Zeit abgetastet und als Zielspannung für die Einstellung der Brennerhöhe verwendet.
2. Wenn **Use Auto Volts** nicht aktiviert ist, wird die Spannung, die als Cut Volts im Abschnitt MATERIAL der PARAMETERS Tab angezeigt wird, als Zielspannung zur Einstellung der Brennerhöhe verwendet.

G-code THC

THC kann direkt vom G-Code aus deaktiviert und aktiviert werden, sofern die THC nicht in der THC-Sektion des Haupt-Registerkarte deaktiviert ist, indem der **motion.digital-out-02** Pin mit den M-Codes M62-M65 gesetzt oder zurückgesetzt wird:

- `M62 P2` deaktiviert THC (synchronisiert mit Bewegung)
- `M63 P2` aktiviert THC (synchronisiert mit Bewegung)
- `M64 P2` schaltet THC (sofort) aus
- `M65 P2` aktiviert THC (sofort)

Es ist wichtig, den Unterschied zwischen **Synchronisiert mit Bewegung** und **Sofort** gründlich zu verstehen:

- `M62` und `M63` (Synchronisiert mit Bewegung) - Die tatsächliche Änderung des angegebenen Ausgangs (z. B. P2 (THC)) erfolgt zu Beginn des nächsten Bewegungsbefehls. Wenn es keinen nachfolgenden Bewegungsbefehl gibt, werden die Ausgangsänderungen nicht ausgeführt. Es empfiehlt sich, einen Bewegungscode (z. B. G0 oder G1) direkt nach einem M62 oder M63 zu programmieren.
- `M64` und `M65` (Sofort) - Diese Befehle werden sofort nach ihrem Empfang durch den Motion Controller ausgeführt. Da sie nicht mit der Bewegung synchronisiert sind, unterbrechen sie das Blending. Das heißt, wenn diese Codes inmitten von aktiven Bewegungscodes verwendet werden, wird die Bewegung angehalten, um diese Befehle zu aktivieren.

Geschwindigkeitsbasierte THC

Wenn die Schnittgeschwindigkeit unter einen bestimmten Prozentsatz von **CutFeedRate** fällt (wie durch den Wert VAD Threshold % im THC-Rahmen des Abschnitts CONFIGURATION auf der Registerkarte PARAMETER definiert), wird die THC gesperrt, bis die Schnittgeschwindigkeit wieder mindestens 99,9 % von **CutFeedRate** beträgt. Dies wird durch das Aufleuchten der Anzeige **VELOCITY ANTI DIVE** im CONTROL Panel auf der Haupt-Registerkarte angezeigt.

Die geschwindigkeitsabhängige THC verhindert, dass die Brennerhöhe verändert wird, wenn die Geschwindigkeit für eine scharfe Ecke oder ein kleines Loch reduziert wird.

Es ist wichtig zu beachten, dass Geschwindigkeits-Reduktion die geschwindigkeitsbasierte THC auf folgende Weise beeinflusst:

1. Wenn die Geschwindigkeitsreduzierung in der Mitte des Schnitts aufgerufen wird, dann wird die THC gesperrt.
2. Die THC bleibt gesperrt, bis die Geschwindigkeitsreduzierung aufgehoben wird, indem sie auf einen Wert oberhalb des **VAD-Schwellenwerts** zurückgesetzt wird, und der Brenner tatsächlich 99,9 % der

CutFeedRate (Vorschubgeschwindigkeit beim Schneiden) erreicht.

Fräserkompensation

LinuxCNC (QtPlasmaC) hat die Fähigkeit, den Schnittpfad des aktuellen Programms automatisch um den Betrag anzupassen, der in der Schnittfugenbreite der Schnittparameter des ausgewählten Materials angegeben ist. Dies ist hilfreich, wenn der G-Code auf den nominalen Schnittpfad programmiert ist und der Benutzer das Programm auf Materialien unterschiedlicher Dicke ausführt, um eine konsistente Größe der Teile zu gewährleisten.

Für die Verwendung der Messerkompensation muss der Benutzer G41.1, G42.1 und G40 mit dem HAL-Stift für die Schnittspaltbreite verwenden:

- `G41.1 D#<_hal[plasmac.kerf-width]>` : Versetzt den Brenner nach links vom programmierten Pfad
- `G42.1 D#<_hal[plasmac.kerf-width]>` : Verschiebt den Brenner nach rechts vom programmierten Pfad
- `G40` schaltet den Schneideausgleich aus

IMPORTANT	WENN DIE **SCHNEIDERKOMPENSATION** IN KRAFT IST, KÖNNEN **G-CODE THC**, **VELOCITY BASED THC** UND **OVER CUT** NICHT VERWENDET WERDEN; ES WIRD EINE FEHLERMELDUNG ANGEZEIGT.

Überspringen der anfänglichen Höhenmessung (Initial Height Sense, IHS)

Initial Height Sense kann auf zwei verschiedene Arten übersprungen werden:

1. Wenn die THC deaktiviert ist oder die THC zwar aktiviert, aber nicht aktiv ist, wird der IHS-Sprung ausgelöst, wenn der Beginn des Schnitts weniger als **Skip IHS** von der letzten erfolgreichen Messung entfernt ist.
2. Wenn die THC aktiviert und aktiv ist, wird der IHS-Sprung durchgeführt, wenn der Beginn des Schnitts weniger als **Skip IHS** vom Ende des letzten Schnitts entfernt ist.

Ein Wert von Null für **Skip IHS** deaktiviert das Überspringen von IHS.

Treten während eines Schnitts Fehler auf, wird das Überspringen von IHS für den nächsten Schnitt deaktiviert, wenn **Skip IHS** aktiviert ist.

Sondieren

Die Abtastung kann entweder mit ohmscher Abtastung oder mit einem Schwimmerschalter erfolgen. Es ist auch möglich, die beiden Methoden zu kombinieren, wobei der Schwimmerschalter einen Rückgriff auf die ohmsche Abtastung ermöglicht. Eine Alternative zur ohmschen Abtastung ist Offset Probing

Wenn der Brenner der Maschine die ohmsche Abtastung nicht unterstützt, kann der Benutzer eine separate Sonde neben dem Brenner verwenden. In diesem Fall würde der Benutzer den Messtaster unterhalb des Brenners ausfahren. Der Messtaster darf NICHT weiter als die minimale Schnitthöhe

unter dem Brenner reichen, und der Abstand der Z-Achse muss als **Ohmscher Versatz** im Rahmen PROBING des Abschnitts CONFIGURATION auf der Registerkarte PARAMETER eingegeben werden.

Die Einrichtung der Sondierung erfolgt im Rahmen PROBING des Abschnitts CONFIGURATION der Registerkarte PARAMETER.

QtPlasmaC kann mit der vollen Geschwindigkeit der Z-Achse sondieren, solange die Maschine genügend Bewegung im Schwimmerschalter hat, um einen eventuellen Überlauf aufzufangen. Wenn der Weg des Schwimmerschalters der Maschine geeignet ist, kann der Benutzer die Sondenhöhe auf einen Wert in der Nähe des MINIMUM_LIMIT der Z-Achse einstellen und die gesamte Abtastung mit voller Geschwindigkeit durchführen.

Einige Schwimmerschalter können eine große Schalthysterese aufweisen, die sich in der Abtastsequenz als übermäßige Zeit bis zum Abschluss der letzten Abtastung bemerkbar macht.

- Diese Zeit kann verkürzt werden, indem man die Geschwindigkeit der letzten Sonde nach oben ändert.
- Diese Geschwindigkeit ist standardmäßig auf 0,001 mm (0,000039") pro Servozyklus eingestellt.
- Es ist möglich, diese Geschwindigkeit um bis zu einem Faktor 10 zu erhöhen, indem die folgende Zeile in die Datei custom.hal eingefügt wird:

```
setp plasmac.probe-final-speed n
```

wobei *n* ein Wert von 1-10 ist. Es wird empfohlen, diesen Wert so niedrig wie möglich zu halten.

Die Verwendung dieser Funktion führt zu einer geringfügigen Änderung der endgültigen Höhe und erfordert eine gründliche Prüfung der Sonde, um die endgültige Höhe zu bestätigen.

Dieser Geschwindigkeitswert wirkt sich auf ALLE Antastungen aus. Wenn der Benutzer also ohmsche Antastungen verwendet und diesen Geschwindigkeitswert ändert, muss er einen Antasttest durchführen, um den erforderlichen Offset einzustellen, um diese Geschwindigkeitsänderung sowie den Schwimmerweg zu kompensieren.

Die Zuverlässigkeit dieser Funktion ist nur so gut wie die Wiederholgenauigkeit des Schwimmerschalters.

NOTE	Sondenhöhe bezieht sich auf die Höhe über der Z-Achse MINIMUM_ LIMIT.

Offset-Sondierung

Bei der Offset-Sondierung wird eine Sonde verwendet, die gegenüber dem Brenner versetzt ist. Diese Methode ist eine Alternative zum ohmschen Antasten und verwendet den Ausgangspin "plasmac.ohmic-enable" zur Betätigung eines Magneten zum Aus- und Einfahren der Sonde. Der Eingangsstift "plasmac.ohmic-probe" wird zur Erkennung des Materials verwendet, und der **Ohmic Offset** im Rahmen PROBING des Abschnitts CONFIGURATION der Registerkarte PARAMETER wird zur Einstellung der richtigen Messhöhe verwendet.

Bei der Sonde kann es sich um eine mechanisch ausfahrbare Sonde, einen fest montierten

Näherungssensor oder auch nur um ein steifes Stück Draht handeln, das etwa 0,5 mm unterhalb der Brennerspitze verläuft. Wenn die Sonde mechanisch ausgefahren wird, muss sie ziemlich schnell aus- und eingefahren werden, um übermäßige Antastzeiten zu vermeiden, und wird in der Regel pneumatisch betrieben.

Um diese Funktion verwenden zu können, muss der Benutzer den Versatz der Sonde von der Brennermitte aus einstellen, indem er das unter Peripherie-Offsets beschriebene Verfahren befolgt.

Um die Offsets manuell zu ändern, kann der Benutzer eine oder beide der folgenden Optionen im Abschnitt **[OFFSET_PROBING]** der <machine_name>Datei __.prefs bearbeiten:

```
X axis = n.n
Y axis = n.n
Delay = t.t
```

wobei *n.n* der Versatz der Sonde von der Brennermitte in Maschineneinheiten für die X- und Y-Achse und *t.t* die Zeit in Sekunden ist, die für eine eventuelle mechanische Ausbringung der Sonde benötigt wird.

Jeder dieser Parameter ist optional und kann auch in beliebiger Reihenfolge angegeben werden. Wird ein Parameter nicht erkannt, ist der Standardwert 0,0. Nach dem X oder Z darf kein Leerzeichen stehen, Kleinschreibung ist zulässig.

Wenn diese Variable in der_<Maschinenname>_.prefs-Datei erscheint und entweder X oder Y ungleich null ist, wird QtPlasmaC **alle** ohmschen Sondierungen als Offset-Probing durchführen. Wenn Offset-Probing gültig ist, kann die Vorschubgeschwindigkeit, mit der sich die X- und Y-Achsen zur Offset-Position bewegen, über den Parameter **Offset Speed** im PROBING-Frame des Einstellungen-Tabs (in der engl. Version "parameters") angepasst werden.

Wenn eine Sondierungssequenz begonnen hat, wird der Pin `plasmac.ohmic-enable` auf "True" gesetzt, wodurch die Sonde ausgefahren wird. Wenn das Material erkannt wird, so wird der Pin `plasmac.ohmic-enable` auf "false" zurückgesetzt, wodurch die Sonde zurückgezogen wird.

Die Sonde beginnt mit der Bewegung zur Offset-Position, während sich die Z-Achse auf die Sondenhöhe absenkt. Die Sondenfahrt beginnt erst, wenn der Einsatzzeitgeber abgelaufen ist. Es ist erforderlich, dass die **Sondenhöhe** im Rahmen PROBING im Abschnitt CONFIGURATION des PARAMETER-Registerkarte über der Oberseite des Materials liegt, um sicherzustellen, dass die Sonde vor der endgültigen vertikalen Abwärtsbewegung der Sonde vollständig in die korrekte X/Y-Position versetzt wird.

IMPORTANT	DIE SONDENHÖHE MUSS FÜR DIE OFFSET-SONDIERUNG ÜBER DER OBERSEITE DES MATERIALS EINGESTELLT WERDEN.

Schnittarten (engl. cut types)

QtPlasmaC erlaubt zwei verschiedene Schnittmodi:

1. **NORMAL CUT** - führt das geladene G-Code-Programm zum Einstechen und Schneiden aus.
2. **NUR DURCHSTECHEN** - durchsticht das Material nur an jeder Startposition, nützlich vor einem

NORMALEN SCHNITT bei << plasma:thick-materials,starken Materialien>>

Es gibt zwei Möglichkeiten, diese Funktion zu aktivieren:

1. Verwenden des Standard custom user buttons, um zwischen den Schnittarten zu wechseln.
2. Hinzufügen der folgenden Zeile zum G-Code-Programm vor dem ersten Schnitt, um den **Pierce Only** -Modus für die aktuelle Datei zu aktivieren:

```
#<pierce-only> = 1
```

Wenn eine benutzerdefinierte Schaltfläche verwendet wird, lädt QtPlasmaC die Datei automatisch neu, wenn die Schnittart umgeschaltet wird.

Löcher schneiden - Intro

Es wird empfohlen, dass die zu schneidenden Löcher einen Durchmesser haben, der mindestens anderthalbmal so groß ist wie die Dicke des zu schneidenden Materials.

Es wird auch empfohlen, dass Löcher mit einem Durchmesser von weniger als 32 mm (1,26") mit 60 % des für Profilschnitte verwendeten Vorschubs geschnitten werden. Dies sollte die THC auch aufgrund von Geschwindigkeitsbegrenzungen ausschließen.

QtPlasmaC kann G-Code-Befehle verwenden, die normalerweise von einem CAM-Postprozessor (PP) eingestellt werden, um das Schneiden von Löchern zu unterstützen. Wenn der Benutzer keinen PP hat oder der PP des Benutzers diese Methoden nicht unterstützt, kann QtPlasmaC den G-Code automatisch entsprechend anpassen. Dieser automatische Modus ist standardmäßig deaktiviert.

Es gibt drei Methoden zur Verbesserung der Qualität von kleinen Löchern:

1. **Geschwindigkeitsreduzierung** - (engl. velocity eduction) Reduzierung der Geschwindigkeit auf ca. 60 % der **CutFeedRate**.
2. **Arc Dwell (Pause At End)** - Der Brenner wird am Ende der Bohrung für kurze Zeit eingeschaltet, während die Bewegung angehalten wird, damit der Lichtbogen aufholen kann.
3. **Overcut** - Schalten Sie den Brenner am Ende des Lochs aus und gehen Sie dann weiter den Weg entlang.

NOTE	Wenn sowohl **Arc Dwell** als auch **Over Cut** gleichzeitig aktiv sind, hat **Over Cut** Vorrang.

IMPORTANT	DIE FUNKTION **OVER CUT** KANN NICHT VERWENDET WERDEN, WENN DIE SCHNEIDEKOMPENSATION AKTIVIERT IST; ES WIRD EINE FEHLERMELDUNG ANGEZEIGT.

Löcher schneiden

G-Code-Befehle können entweder von einem CAM-Postprozessor (PP) oder durch Handcodierung erstellt werden.

Reduzierung der Lochschnittgeschwindigkeit

Wenn das Schneiden eines Lochs eine reduzierte Geschwindigkeit erfordert, würde der Benutzer den folgenden Befehl verwenden, um die Geschwindigkeit einzustellen: `M67 E3 Qnn` wobei nn der Prozentsatz der gewünschten Geschwindigkeit ist. Zum Beispiel würde `M67 E3 Q60` die Geschwindigkeit auf 60% der **CutFeedRate** des aktuellen Materials einstellen.

Um diese Funktion nutzen zu können, muss die adaptive Vorschubsteuerung (M52) von LinuxCNC eingeschaltet sein (P1). Das ist auch eine Voraussetzung für Paused Motion während der Cut Recovery.

Um **Loch Schneiden Geschwindigkeits-Verringerung** (engl. Hole Cutting Velocity Reduction) zu aktivieren, muss die Präambel des G-Codes die folgende Zeile enthalten:

```
M52 P1
```

Um **Loch Schneiden Geschwindigkeits-Reduktion** (engl. Hole Cutting Velocity Reduction) an einem beliebigen Punkt zu deaktivieren, verwenden Sie den folgenden Befehl:

```
M52 P0
```

Siehe Abschnitt Geschwindigkeitsbasiertes THC (engl. Velocity Based THC).

Beispielcode für das Schneiden von Löchern mit reduzierter Geschwindigkeit.

```
G21 (metrisch)
G64 P0.005
M52 P1 (aktivieren adaptiven Vorschubs)
F#<_hal[plasmac.cut-feed-rate]> (Vorschubrate aus Schnittparametern)
G0 X10 Y10
M3 $0 S1 (Startschnitt)
G1 X0
M67 E3 Q60 (Vorschubgeschwindigkeit auf 60% reduzieren)
G3 I10 (das Loch)
M67 E3 Q0 (Wiederherstellung der Vorschubgeschwindigkeit auf 100%)
M5 $0 (Endschnitt)
G0 X0 Y0
M2 (Endauftrag)
```

Arc Dwell (Pause am Ende)

Diese Methode kann aufgerufen werden, indem der Parameter Pause At End im Rahmen MATERIAL der Registerkarte PARAMETER festgelegt wird.

Überschnitt *(engl. overcut)*

Der Brenner kann am Ende der Bohrung ausgeschaltet werden, indem der Pin `motion.digital-out-03` mit den M-Codes `M62` (Synchronized with Motion)* oder `M64` (Immediate)* gesetzt wird. Nach dem Ausschalten des Brenners muss der Brenner wieder eingeschaltet werden, bevor mit dem nächsten Schnitt begonnen wird, indem der Pin `motion.digital-out-03` mit den M-Codes `M63` oder `M65` zurückgesetzt wird. Dies geschieht automatisch durch den QtPlasmaC G-Code Parser, wenn er einen M5-Befehl erreicht, ohne einen `M63 P3` oder `M65 P3` zu sehen.

Nachdem der Brenner ausgeschaltet wurde, wird der Lochpfad für eine Standardlänge von 4 mm (0,157

Zoll) verfolgt. Dieser Abstand kann durch Hinzufügen von `#<oclength> = n` zur G-Code-Datei angegeben werden.

- `M62 P3` schaltet den Brenner aus (synchronisiert mit der Bewegung)
- `M63 P3` ermöglicht das Einschalten des Brenners (synchronisiert mit der Bewegung)
- `M64 P3` schaltet den Brenner aus (sofort)
- `M65 P3` wird erlauben, den Brenner einzuschalten (sofort)

Es ist wichtig, den Unterschied zwischen **synchronisiert mit Bewegung** und **unmittelbar** genau zu verstehen:

- `M62` und `M63` (Synchronisiert mit Bewegung) - Die tatsächliche Änderung des angegebenen Ausgangs (z. B. P2 (THC)) erfolgt zu Beginn des nächsten Bewegungsbefehls. Wenn es keinen nachfolgenden Bewegungsbefehl gibt, werden die Ausgangsänderungen nicht ausgeführt. Es empfiehlt sich, einen Bewegungscode (z. B. G0 oder G1) direkt nach einem M62 oder M63 zu programmieren.
- `M64` und `M65` (Sofort) - Diese Befehle werden sofort nach ihrem Empfang durch den Motion Controller ausgeführt. Da sie nicht mit der Bewegung synchronisiert sind, unterbrechen sie das Blending. Das heißt, wenn diese Codes inmitten von aktiven Bewegungscodes verwendet werden, wird die Bewegung angehalten, um diese Befehle zu aktivieren.

Beispiel-Code:

```
G21 (metrisch)
G64 P0.005
M52 P1 (aktivieren von adaptivem Vorschub)
F#<_hal[plasmac.cut-feed-rate]> (Vorschubrate aus Schnittparametern)
G0 X10 Y10
M3 $0 S1 (Startschnitt)
G1 X0
M67 E3 Q60 (Vorschubgeschwindigkeit auf 60% reduzieren)
G3 I10 (das Loch)
M62 P3 (Taschenlampe ausschalten)
G3 X0.8 Y6.081 I10 (Fortbewegung für 4 mm)
M63 P3 (Taschenlampe einschalten lassen)
M67 E3 Q0 (Wiederherstellung der Vorschubgeschwindigkeit auf 100%)
M5 $0 (Endschnitt)
G0 X0 Y0
M2 (Endauftrag)
```

Löcher schneiden - automatisch

QtPlasmaC hat die Fähigkeit, den G-Code automatisch zu modifizieren, um die Geschwindigkeit zu reduzieren und/oder **Überschnitt** (engl. over cut) anzuwenden, was beim Schneiden von Löchern nützlich sein kann.

Für eine gültige Lochabtastung ist es erforderlich, dass alle Werte in der G2 oder G3 G-Code-Zeile explizit sind, ein Fehlerdialog wird angezeigt, wenn irgendwelche Werte mathematisch berechnet werden.

QtPlasmaC Loch-Erkennung (engl. hole sensing) ist standardmäßig deaktiviert. Sie kann mit den

folgenden G-Code-Parametern aktiviert/deaktiviert werden, um den gewünschten Lochabtastungsmodus auszuwählen:

- `#<holes> = 0` - Veranlasst QtPlasmaC, die Locherkennung zu deaktivieren, wenn sie zuvor aktiviert war.
- `#<holes> = 1` - Veranlasst QtPlasmaC, die Geschwindigkeit von Löchern unter 32 mm (1.26") auf 60% von **CutFeedRate** zu reduzieren.
- `#<holes> = 2` - Bewirkt, dass QtPlasmaC Überschnitt das Loch zusätzlich zu den Geschwindigkeitsänderungen in Einstellung 1 ändert.
- `#<holes> = 3` - Veranlasst QtPlasmaC, die Geschwindigkeit von Löchern unter 32 mm (1.26") und Bögen unter 16 mm (0.63") auf 60% der **CutFeedRate** zu reduzieren.
- `#<holes> = 4` - Bewirkt, dass QtPlasmaC Over cut das Loch zusätzlich zur Geschwindigkeitsänderung in Einstellung 3.

Die Standard-Lochgröße für die QtPlasmaC-Lochabtastung ist 32&8239;mm (1.26"). Es ist möglich, diesen Wert mit dem folgenden Befehl in einer G-Code-Datei zu ändern:

- `#<h_diameter> =` *nn* - Um einen Durchmesser (*nn*) im gleichen Einheitensystem wie für den Rest der G-Code-Datei festzulegen.

Die Standardgeschwindigkeit für kleine Löcher in QtPlasmaC beträgt 60% der aktuellen Vorschubgeschwindigkeit. Es ist möglich, diesen Wert mit dem folgenden Befehl in einer G-Code-Datei zu ändern:

- `#<h_velocity> =` *nn* - um den Prozentsatz (*nn*) der aktuellen Vorschubgeschwindigkeit einzustellen.

Überschnitt *(engl. overcut)*

Wenn die Hole Sensing-Modi 2 oder 4 aktiv sind, überschneidet QtPlasmaC das Loch zusätzlich zu den Geschwindigkeitsänderungen, die mit den Modi 1 und 3 verbunden sind.

Die standardmäßige Überschnittlänge für die QtPlasmaC-Lochabtastung beträgt 4 mm (0,157"). Es ist möglich, diesen Wert mit dem folgenden Befehl in einer G-Code-Datei zu ändern:

- `#<oclength> =` *nn* zur Angabe einer Überschnittlänge (*nn*) im gleichen Einheitensystem wie der Rest der G-Code-Datei.

Arc Dwell (Pause am Ende)

Diese Funktion kann zusätzlich zur Einstellung des gewünschten Lochabtastmodus über den entsprechenden G-Code-Parameter verwendet werden, indem der Parameter Pause At End im MATERIAL-Rahmen der PARAMETER Registerkarte gesetzt wird.

Beispiel-Code:

```
G21 (metric)
G64 P0.005
M52 P1 (aktivieren von adaptiven Vorschub)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
#<holes> = 2 (over cut for holes)
```

```
#<oclength> = 6.5 (optional, 6.5 mm over cut length)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
G3 I10 (the hole)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

NOTE Es ist in Ordnung, mehrere und gemischte Lochbefehle in einer einzigen G-Code-Datei zu haben.

Einzelschnitt

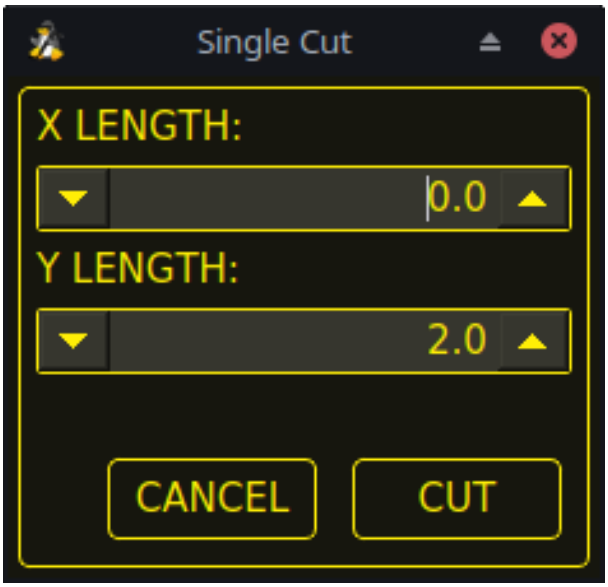
Ein Einzelschnitt (engl. single cut) ist eine einzelne unidirektionale Schnittbewegung, die häufig verwendet wird, um ein Blech in kleinere Teile zu schneiden, bevor ein G-Code-Programm ausgeführt wird.

Die Maschine muss vor Beginn eines einzelnen Einzelschnitts referenziert werden.

Ein Einzelschnitt wird von der aktuellen X/Y-Position der Maschine aus gestartet.

Automatischer Einzelschnitt

Dies ist die bevorzugte Methode. Die Parameter für diese Methode werden in das folgende Dialogfeld eingegeben, das nach dem Drücken einer user button angezeigt wird, die für den Einzelschnitt programmiert wurde:



1. Joggen Sie zur gewünschten X/Y-Startposition.
2. Stellen Sie das gewünschte Material ein, oder bearbeiten Sie die Vorschubgeschwindigkeit für das Standardmaterial auf der Registerkarte PARAMETERS-tab.
3. Drücken Sie die zugewiesene Benutzertaste für den Einzelschnitt.
4. Geben Sie die Schnittlänge entlang der X- und/oder Y-Achse ein.

5. Drücken Sie die CUT-Taste und der Schnitt beginnt.

Pendant Einzelschnitt

Wenn die Maschine mit einem Steuergerät (engl. pendant) ausgestattet ist, das die Spindel starten und stoppen sowie die X- und Y-Achsen verfahren kann, dem Benutzer ermöglichend, einen einzelnen Schnitt manuell durchführen.

1. Joggen Sie zur gewünschten X/Y-Startposition.
2. Stellen Sie die gewünschte Vorschubgeschwindigkeit mit dem Schieberegler Jog Speed ein.
3. Starten Sie den Schnittvorgang, indem Sie die Spindel einschalten.
4. Nach dem Sondieren wird der Brenner gezündet.
5. Wenn der Lichtbogen OK ist, kann die Maschine mit den Jog-Tasten entlang der Schnittlinie verfahren werden.
6. Wenn der Schnitt beendet ist, stoppen Sie die Spindel.
7. Der Brenner schaltet sich aus und die Z-Achse kehrt in die Ausgangsposition zurück.

Manueller Einzelschnitt

Für den manuellen Einzelschnitt muss entweder Tastaturkürzel im Abschnitt GUI SETTINGS der Registerkarte EINSTELLUNGEN aktiviert sein, oder eine benutzerdefinierte Schaltfläche muss als Manueller Schnitt-Schaltfläche angegeben sein.

Wenn der Benutzer eine benutzerdefinierte Taste verwendet, ersetzen Sie **F9** durch **Benutzertaste** in der folgenden Beschreibung.

1. Joggen Sie zur gewünschten X/Y-Startposition.
2. Starten Sie den Vorgang durch Drücken von **F9**. Die Tippgeschwindigkeit wird automatisch auf die Vorschubgeschwindigkeit des aktuell ausgewählten Materials eingestellt. Das Jog-Etikett blinkt, um anzuzeigen, dass die Jog-Geschwindigkeit vorübergehend außer Kraft gesetzt ist (die Manipulation der Jog-Geschwindigkeit ist deaktiviert, während ein manueller Schnitt aktiv ist). Das Symbol **CYCLE START** wechselt zu **MANUAL CUT** und blinkt.
3. Nach dem Sondieren wird der Brenner gezündet.
4. Wenn der Lichtbogen OK ist, kann die Maschine mit den Jog-Tasten entlang der Schnittlinie verfahren werden.
5. Die Z-Höhe bleibt für die Dauer des manuellen Schnitts auf der Schnitthöhe fixiert, unabhängig vom Status der Brennerhöhensteuerung **ENABLE**.
6. Wenn der Schnitt komplett ist drücken F9 oder Esc oder die **CYCLE STOP** Taste.
7. Der Brenner schaltet sich aus und die Z-Achse kehrt in die Ausgangsposition zurück.
8. Die Jog-Geschwindigkeit wird automatisch auf den Wert zurückgesetzt, den sie vor Beginn des manuellen Schnittvorgangs hatte, das Etikett hört auf zu blinken und die Jog-Speed-Manipulation wird aktiviert. **MANUAL CUT** hört auf zu blinken und kehrt zu **CYCLE START** zurück.

NOTE	Wenn der Brenner während des Schneidens abfackelt, muss der Benutzer trotzdem **F9** oder **Esc** oder die Taste **CYCLE STOP** drücken, um den Schnitt zu beenden. Dadurch

werden die Z-Offsets gelöscht und der Brenner kehrt in die Ausgangsposition zurück.

Dicke Materialien

Das Schneiden dicker Materialien kann insofern problematisch sein, als die große Menge geschmolzenen Metalls, die beim Einstechen entsteht, die Lebensdauer der Verschleißteile verkürzen kann und außerdem eine Pfütze verursachen kann, die so groß ist, dass der Brenner beim Erreichen der Schnitthöhe auf die Pfütze treffen kann.

Im QtPlasmaC sind mehrere Funktionen integriert, um diese Probleme zu entschärfen: „Pierce Only“ (engl. für "nur durchstechen") und „Puddle Jump“ (engl. für Pfützen springen), die in diesem Abschnitt beschrieben werden, sowie „Wiggle Pierce“ und „Ramp Pierce“, die im Abschnitt Moving Pierce erläutert sind.

Nur Durchstechen *(engl. pierce only)*

Der Modus **Pierce Only** (engl. für "nur durchstechen") konvertiert das geladene G-Code-Programm und führt dann das Programm aus, um das Material an der Startposition jedes Schnitts zu lochen. Ritz- und Punktierbefehle werden ignoriert, und es findet kein Lochstechen an diesen Stellen statt.

Dieser Modus ist nützlich für dicke Materialien, die beim Durchstechen genügend Schlacke auf der Materialoberfläche erzeugen können, um den Brenner beim Schneiden zu stören. Das gesamte Blech kann durchstochen und dann vor dem Schneiden gereinigt werden.

Es ist möglich, Verbrauchsmaterialien, die kurz vor dem Ende ihrer Lebensdauer stehen, zum Durchstechen zu verwenden und sie dann gegen gute Verbrauchsmaterialien auszutauschen, die beim Schneiden verwendet werden.

Die Position des Einstichs im **Pierce Only**-Modus kann in der X- und/oder Y-Achse versetzt werden, um sicherzustellen, dass der Lichtbogen beim Einstich nach dem Zurückkehren in den **Normal Cut**-Modus korrekt übertragen werden kann. Die Parameter für die X- und Y-Versätze befinden sich im Abschnitt PIERCE ONLY des KONFIGURATION-Bereichs der PARAMETER-Registerkarte.

Pierce Only (engl. für "nur durchstechen") ist eine von zwei verschiedenen Cut-Typen

Pfützensprung *(engl. puddle jump)*

Der **Pfützensprung** ist die Höhe, auf die sich der Brenner nach dem Einstechen und vor dem Erreichen der **Schnitthöhe** bewegt, und wird als Prozentsatz der **Einstechhöhe** angegeben. Dadurch kann der Brenner jede Pfütze aus geschmolzenem Material, die durch das Einstechen verursacht werden kann, entfernen. Die maximal zulässige Höhe beträgt 200% der **Lochstechhöhe** (engl. pierce height).

Die Einstellungen für den **Puddle Jump** sind in Schnittparameter beschrieben.

Die empfohlene Option ist **Pierce Only** (engl. für "nur durchstechen"), da sie fast verbrauchte Verbrauchsmaterialien verwenden kann.

IMPORTANT	PFÜTZENSPRUNG (engl. **puddle jump**) IST WÄHREND DER WIEDERHERSTELLUNG DES SCHNITTS DEAKTIVIERT

Mesh-Modus (Streckmetallschneiden, engl. Expanded Metal Cutting)

QtPlasmaC ist in der Lage, Streckmetall zu schneiden, vorausgesetzt, die Maschine verfügt über einen Pilotlichtbogenbrenner und kann im CPA-Modus (Constant Pilot Arc) betrieben werden.

Der **Mesh Mode** deaktiviert die THC und ignoriert auch ein verlorenes Arc OK-Signal während eines Schnitts. Er kann durch Aktivieren der Schaltfläche **Mesh Mode** im Abschnitt CONTROL auf der Registerkarte MAIN ausgewählt werden.

Wenn die Maschine über eine RS485-Kommunikation mit einem Hypertherm PowerMax-Plasmaschneider verfügt, wird durch die Auswahl von **Mesh Mode** automatisch der **Cut Mode** für das aktuell ausgewählte Material außer Kraft gesetzt und auf Cut Mode 2 (CPA) eingestellt. Wenn **Mesh Mode** deaktiviert ist, wird der **Cut Mode** auf den Standard-Schneidmodus für das aktuell ausgewählte Material zurückgesetzt.

Es ist auch möglich, einen Schnitt im **Mesh-Modus** zu starten, ohne ein Bogen-OK-Signal zu erhalten, indem Sie die Schaltfläche **Bogen-OK ignorieren** im Abschnitt KONTROLLE auf der Registerkarte MAIN aktivieren.

Sowohl **Mesh Mode** als auch **Ignore Arc OK** können jederzeit während eines Auftrags aktiviert/deaktiviert werden.

Bogen (engl. arc) ignorieren OK

Ignore Arc OK Modus deaktiviert das THC, beginnt einen Cut, ohne dass ein Arc OK-Signal erforderlich ist, und ignoriert ein verlorenes Arc OK-Signal während eines Cuts.

Dieser Modus kann ausgewählt werden durch:

1. Aktivieren Sie die Schaltfläche **Ignore Arc OK** im Abschnitt CONTROL der HAUPTREGISTERKARTE.
2. HAL-Pin **motion.digital-out-01** per G-Code auf 1 setzen.
   - `M62 P1` aktiviert **Ignore Arc OK** (synchronisiert mit Bewegung)
   - `M63 P1` deaktiviert **Ignore Arc OK** (synchronisiert mit Bewegung)
   - `M64 P1` aktiviert **Ignore Arc OK** (Sofort)
   - `M65 P1` deaktiviert **Ignore Arc OK** (Sofort)

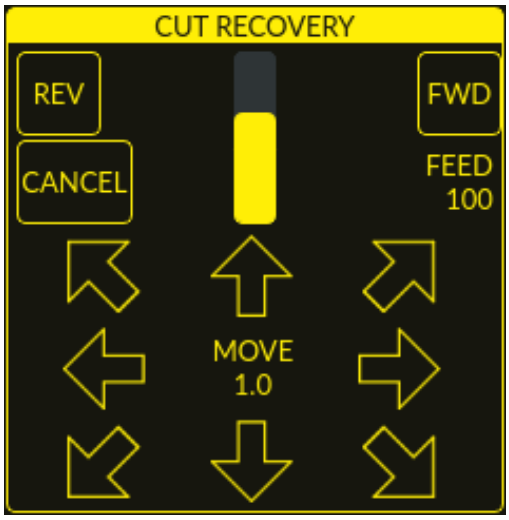
Es ist wichtig, den Unterschied zwischen **synchronisiert mit Bewegung** und **unmittelbar** genau zu verstehen:

- `M62` und `M63` (Synchronisiert mit Bewegung) - Die tatsächliche Änderung des angegebenen Ausgangs (z. B. P2 (THC)) erfolgt zu Beginn des nächsten Bewegungsbefehls. Wenn es keinen nachfolgenden Bewegungsbefehl gibt, werden die Ausgangsänderungen nicht ausgeführt. Es empfiehlt sich, einen Bewegungscode (z. B. G0 oder G1) direkt nach einem M62 oder M63 zu programmieren.
- `M64` und `M65` (Sofort) - Diese Befehle werden sofort nach ihrem Empfang durch den Motion Controller ausgeführt. Da sie nicht mit der Bewegung synchronisiert sind, unterbrechen sie das Blending. Das heißt, wenn diese Codes inmitten von aktiven Bewegungscodes verwendet werden, wird die Bewegung angehalten, um diese Befehle zu aktivieren.

Dieser Modus kann auch in Verbindung mit dem **Mesh-Modus** verwendet werden, wenn der Benutzer kein Arc OK-Signal benötigt, um den Schnitt zu starten.

Sowohl **Mesh Mode** als auch **Ignore Arc OK** können jederzeit während eines Auftrags aktiviert/deaktiviert werden.

Schnitt Wiederaufnahme (engl. cut recovery)



Diese Funktion erzeugt eine Tafel CUT RECOVERY, die es ermöglicht, den Brenner während einer unterbrochenen Bewegung (engl. paused motion)-Ereignisses von der Schneidbahn wegzubewegen, um den Brenner über einem Reststück des zu schneidenden Materials zu positionieren, so dass der Schnitt mit einem minimierten Lichtbogen-Divot neu beginnt. Das Feld CUT RECOVERY wird automatisch über dem Feld JOGGING angezeigt, wenn die Bewegung angehalten wird.

Es ist vorzuziehen, die Brennerposition von dem Punkt aus anzupassen, an dem die angehaltene Bewegung aufgetreten ist, aber wenn vor dem Setzen des neuen Startpunkts eine Bewegung entlang des Schnittpfads erforderlich ist, kann der Benutzer die angehaltenen Bewegungssteuerungen (**REV**, **FWD** und ein **JOG-SPEED**-Schieberegler) oben im CUT RECOVERY-Bedienfeld verwenden. Sobald der Benutzer mit der Positionierung des Brenners entlang der Schnittbahn zufrieden ist, erfolgt das Verlassen der Schnittbahn durch Drücken der Tasten **DIRECTION**. Jedes Drücken der **DIRECTION**-Taste bewegt den Brenner um eine Entfernung, die dem Parameter **Schnittfugenbreite** des aktuell ausgewählten Materials entspricht.

In dem Moment, in dem der Brenner aus dem Schneidpfad bewegt wurde, werden die Steuerelemente für die angehaltene Bewegung (**RÜCKW** (engl. rev), **VORW** (engl. fwd) und ein **JOG-GESCHW** (engl. jog-speed)-Schieberegler) oben im Bedienfeld CUT RECOVERY deaktiviert.

Sobald die Brennerposition zufriedenstellend ist, drücken Sie **ZYKLUS-WIEDERAUFNAHME** und der Schnitt wird von der neuen Position aus fortgesetzt und fährt die kürzeste Strecke zur ursprünglichen, angehaltenen Bewegungsposition. Das Feld CUT RECOVERY wird geschlossen und das Feld JOGGING wird angezeigt, wenn der Brenner an die ursprüngliche angehaltene Bewegungsposition zurückkehrt.

Durch Drücken von **CANCEL MOVE** (engl. für Bewebung abbrechen) bewegt sich die Taschenlampe wieder dorthin, wo sie positioniert war, bevor die Richtungstasten verwendet wurden, um den Brenner zu versetzen. Es wird keine **REV** (engl. Abkürzung für rückwärts) oder **FWD** (engl. Abkürzung für

vorwärts) Bewegung zurückgesetzt.

Wenn Sie **CYCLE STOP** drücken, bewegt sich der Brenner zurück in die Position, in der er sich befand, bevor die Richtungstasten zum Versetzen des Brenners verwendet wurden, und das Overlay des CUT RECOVERY-Feldes kehrt zum JOGGING-Feld zurück. Eine **REV**- oder **FWD**-Bewegung wird dadurch nicht zurückgesetzt.

Wenn ein Ausrichtungslaser eingerichtet wurde, ist es möglich, den Laser während der Schnittwiederherstellung für eine sehr genaue Positionierung der neuen Startkoordinaten zu verwenden. Wenn entweder der X-Achsen-Offset oder der Y-Achsen-Offset für den Laser dazu führen würde, dass sich die Maschine außerhalb der Grenzen bewegt, wird eine Fehlermeldung angezeigt.

Um einen Laser für die Schnittwiederherstellung zu verwenden, wenn er während eines Schnitts pausiert:

1. Klicken Sie auf die Schaltfläche **LASER**.
2. Die Schaltfläche **LASER** wird deaktiviert, der HAL-Pin mit dem Namen qtplasmac.laser_on wird eingeschaltet und die X- und Y-Achse werden versetzt, sodass das Laser-Fadenkreuz die Startkoordinaten des Schnitts anzeigt, wenn er fortgesetzt wird.
3. Fortsetzung der Wiederherstellung des Schnittes wie oben beschrieben.

Wenn ein Laser-Offset in Kraft ist und dann **Bewegung abbrechen** gedrückt wird, so wird der Offset ebenfalls gelöscht.

NOTE	Die Bewegungen zur Wiederherstellung des Schnitts (engl. cut recovery movements) sind auf einen Radius von 10 mm ab dem Punkt begrenzt, an dem das Programm angehalten wurde, oder ab dem letzten Punkt auf dem Schnittpfad (engl. cut path), wenn eine angehaltene Bewegung verwendet wurde.

IMPORTANT	PFÜTZENSPRUNG (engl. puddle jump) IST WÄHREND DER WIEDERHERSTELLUNG DES SCHNITTS DEAKTIVIERT

Von Zeile ausführen

Wenn der Benutzer die Option Run From Line im Abschnitt GUI SETTINGS der Registerkarte EINSTELLUNGEN aktiviert hat, kann er mit den folgenden Methoden von jeder Zeile in einem G-Code-Programm aus starten:

1. Anklicken einer beliebigen Zeile im Vorschaufenster
2. Anklicken einer beliebigen Zeile im G-Code-Fenster

Es ist wichtig zu beachten, dass G-Code-Programme mit dieser Methode von jeder ausgewählten Zeile aus ausgeführt werden können, ein Lead-in ist jedoch je nach ausgewählter Zeile möglicherweise nicht möglich. In diesem Fall wird eine Fehlermeldung angezeigt, um den Benutzer darüber zu informieren, dass die Lead-in - Berechnung nicht möglich war.

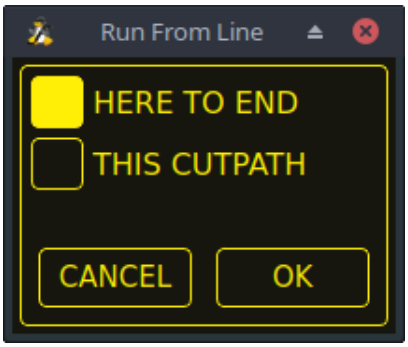
Sobald der Benutzer den Startpunkt ausgewählt hat, blinkt die Schaltfläche **CYCLE START "SELECTED**

nn", wobei *nn* die entsprechende ausgewählte Zeilennummer ist. Wenn Sie auf diese Schaltfläche klicken, wird das folgende Dialogfeld "Run From Line" angezeigt:

Es ist nicht möglich, Run From Line innerhalb eines Unterprogramms zu verwenden. Wenn der Benutzer eine Zeile innerhalb eines Unterprogramms auswählt und auf **"SELECTED *nn*"** klickt, wird eine Fehlermeldung angezeigt, die den O-Code-Namen des Unterprogramms enthält.

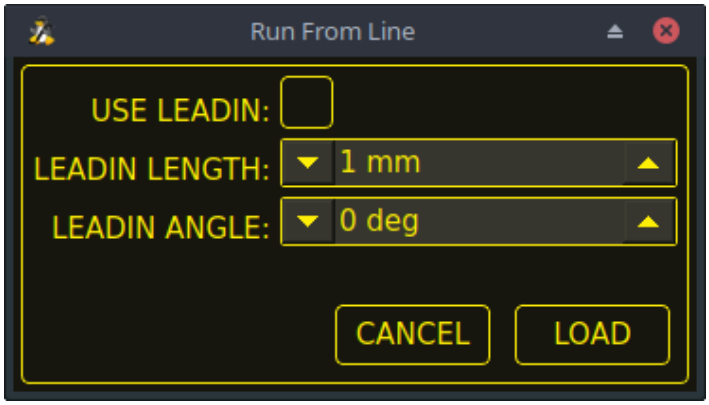
Es ist nicht möglich, Run From Line zu verwenden, wenn der vorherige G-Code die Fräserkompensation aktiviert hat. Wenn der Benutzer eine Linie auswählt, während die Fräserkompensation aktiv ist und auf **"SELECTED *nn*"** klickt, wird eine Fehlermeldung angezeigt.

Es ist möglich, eine neue Zeile auszuwählen, während Run From Line aktiv ist.



HERE TO END

HERE TO END will run from the beginning of the selected line to the end of the G-code file. The user will be presented with the option of adding a lead-in if the selected line falls within an "active" cutting operation (between an M3 and M5).



Name	Beschreibung
USE LEADIN	Mit diesem Optionsfeld kann der Benutzer die ausgewählte Zeile mit einem Lead-in starten.
LEADIN-LÄNGE	Wird USE LEADIN gewählt, so wird die Länge des *lead in* in Maschineneinheiten eingestellt.

Name	Beschreibung
LEADIN-WINKEL	Wenn USE LEADIN ausgewählt ist, wird hiermit der Annäherungswinkel für den Lead-in festgelegt. Der Winkel wird so gemessen, dass positive Wertsteigerungen den Lead-in gegen den Uhrzeigersinn verschieben: 0 Grad = 3-Uhr-Position 90 Grad = 12-Uhr-Position 180 Grad = 9-Uhr-Position 270 Grad = 6-Uhr-Position
ABBRECHEN	Mit diesem Button brechen Sie das Dialogfeld "Von Zeile ausführen" und alle Auswahlen ab.
LADEN (engl. LOAD)	Mit dieser Schaltfläche wird ein temporäres "rfl.ngc"-Programm mit allen ausgewählten Lead-in-Parametern geladen. Wenn der Lead-in für die ausgewählte Zeile nicht berechnet werden kann, wird die folgende Fehlermeldung angezeigt: "Für diesen Schnitt kann kein Lead-in berechnet werden Das Programm wird von der ausgewählten Linie aus laufen, ohne dass ein Lead-in angewendet wird"

Nachdem Sie **LADEN** gedrückt haben, ändert sich die blinkende Schaltfläche "AUSGEWÄHLTE *nn*" (engl. selected) in die Schaltfläche **STARTEN VOM ZEILENZYKLUSSTART**. Klicken Sie auf diese Schaltfläche, um das Programm am Anfang der ausgewählten Zeile zu starten.

THIS CUTPATH

THIS CUTPATH wird nur den Schnittpfad ausführen, zu dem das ausgewählte Segment gehört.

Der blinkende "SELECTED *nn*"-Button wird zum **RUN FROM LINE CYCLE START**-Button wechseln. Klicken Sie auf diesen Button, um den ausgewählten Schnittpfad auszuführen.

Die Auswahl von "Run From Line" kann auf folgende Weise aufgehoben werden:

1. Klicken Sie auf den Hintergrund des Vorschaufensters - diese Methode hebt die Auswahl einer Schnittlinie im Vorschaufenster oder einer G-Code-Zeile im G-Code-Fenster auf.
2. Klicken Sie auf den Text der ersten Zeile des G-Code-Programms in der G-Code-Anzeige - diese Methode hebt die Auswahl einer Schnittlinie im Vorschaufenster oder einer G-Code-Zeile im G-Code-Fenster auf.
3. Klicken auf **RELOAD** in der Kopfzeile des G-Code-Fensters - diese Methode bricht den Prozess "Run From Line" ab, wenn im Dialogfeld "Run From Line" auf LOAD geklickt wurde und "rfl.ngc" als geladener Dateiname in der Kopfzeile des G-Code-Fensters angezeigt wird. Dadurch kehrt der Benutzer zur ursprünglich geladenen Datei zurück.

NOTE Although Run From Line allows the user to begin execution at any line, not every possible scenario can be fully tested. Most testing has focused on recovering typical cutting operations. Therefore when recovering outside of cutting operations, users

> should select a starting line that provides the GUI with the most context about the operation. For example, when restarting a spotting operation, choose the G0 line before the M3 command rather than the M3 itself or the G1 X0.000001 move in the middle of the spotting operation.

Schreiber (engl. scribe)

Zusätzlich zum Plasmabrenner kann mit QtPlasmaC ein Ritzgerät betrieben werden.

Die Verwendung eines Ritzers erfordert die Verwendung der LinuxCNC-Werkzeugtabelle. Tool 0 ist der Plasmabrenner und Tool 1 ist der Ritz zugewiesen. Die Ritz X-und Y-Achsen Offsets aus dem Plasmabrenner müssen in die LinuxCNC Werkzeugtabelle eingegeben werden. Dies geschieht durch Editieren der Werkzeugtabelle über die Haupt-GUI oder durch Editieren der **tool.tbl** Datei im <***Maschinenname***> Konfigurationsverzeichnis. Dies wird getan, nachdem der Ritzer kann auf das Werkstück zu bewegen, um den entsprechenden Offset zu bestimmen.

Die Versätze des Plasmabrenners für X und Y sind immer Null. Die Werkzeuge werden mit dem Befehl **T*n* M6** ausgewählt, gefolgt von einem Befehl **G43 H0**, der erforderlich ist, um die Offsets anzuwenden. Das Werkzeug wird dann mit dem Befehl **M3 $*n* S1** gestartet. Für *n* verwenden Sie 0 für Brennschneiden oder 1 für Anreißen.

Um den Ritzvorgang zu stoppen, verwenden Sie den G-Code-Befehl **M5 $1**.

Hat der Benutzer die HAL-Pins für den Schreiber (engl. scribe) im Konfigurationsassistent noch nicht zugewiesen, so können sie dies tun, indem er den entsprechenden Konfigurations-Assistenten (engl. configuration wizard) nutzt oder durch manuelle Bearbeitung der HAL-Datei, siehe Anpassen von QtPlasmaC.

Es gibt zwei HAL-Ausgangspins, die zum Betreiben des Gravierers verwendet werden; der erste Pin wird verwendet, um den Gravierer zu aktivieren, wodurch der Gravierer an die Oberfläche des Materials bewegt wird. Nachdem die Arm Delay (engl. für Verzögerung des Gravier-Arms) vergangen ist, wird der zweite Pin verwendet, um den Gravierer zu starten. Nachdem die On Delay (engl. für Verzögerung des Starts) vergangen ist, beginnt die Bewegung.

Bei Verwendung von QtPlasmaC muss nach dem Aktivieren des Schreibers entweder der Brenners oder der Schreibers in jeder G-Code-Datei als LinuxCNC-Tool ausgewählt werden.

Der erste Schritt ist die Einstellung der Offsets für die Gravur (engl. scribe), wie unter Peripherie-Offsets beschrieben.

Der letzte Schritt ist die Einstellung der Schreiber Verzögerungen (engl. scribe delays) erforderlich:

1. **Arm Delay** - ermöglicht es dem Schreiber, auf die Oberfläche des Materials abzusteigen.
2. **On Delay** - erlaubt Zeit für den Schreiber zu starten, bevor die Bewegung beginnt.

Speichern Sie die Parameter in der Registerkarte Config.

Nach Abschluss der obigen Anweisungen kann der Schreiber manuell getestet werden, indem ein Befehl **M3 $1 S1** im MDI-Eingang ausgegeben wird. Der Benutzer kann es hilfreich finden, diese Methode zu

verwenden, um eine kleine Delle auszuhöhlen und dann zu versuchen, den Brenner an der gleichen Stelle pulsieren zu lassen, um die Offsets zwischen dem Schreiber und dem Brenner auszurichten.

Um den Schreiber mit G-Code zu nutzen:

```
...
M52 P1 (adaptiven Vorschub aktivieren)
F#<_hal[plasmac.cut-feed-rate]>
T1 M6 (Ritzer auswählen)
G43 H0 (Offsets für aktuelles Werkzeug auswählen)
M3 $1 S1 (Ritzen starten)
.
M5 $1 (Ritzen wird gestoppt)
.
T0 M6 (Brenner auswählen)
G43 H0 (Offsets für aktuelles Werkzeug anwenden)
G0 X0 Y0 (Park-Position)
M5 $-1 (Ende des gesamten Vorgangs)
```

Es empfiehlt sich, am Ende des Programms vor der letzten schnellen Parkbewegung wieder auf den Brenner umzuschalten, damit sich die Maschine im Leerlauf immer im gleichen Zustand befindet.

Der Benutzer kann während eines Programms beliebig oft zwischen dem Brenner und dem Gravierer wechseln, indem er die entsprechenden G-Codes verwendet.

Die Ausgabe von **M3 S1** (ohne $*n*) bewirkt, dass sich die Maschine so verhält, als ob ein **M3 $0 S1** ausgegeben worden wäre, und die Ausgabe von **M5** (ohne $*n*) bewirkt, dass sich die Maschine so verhält, als ob ein **M5 $0** ausgegeben worden wäre. Dies steuert standardmäßig das Abfeuern des Brenners, um die Abwärtskompatibilität mit früheren G-Code-Dateien zu gewährleisten.

WARNING	Wenn in der Datei <*Maschinenname*>.hal ein manueller Werkzeugwechselparameter vorhanden ist, wandelt QtPlasmaC diesen in einen automatischen Werkzeugwechsel um.

Spotting

Um das Material vor dem Bohren usw. zu markieren, kann QtPlasmaC den Brenner kurzzeitig pulsieren, um die zu bohrende Stelle zu markieren.

Spotting kann mit den folgenden Schritten konfiguriert werden:

1. Stellen Sie die Lichtbogenspannung **Schwellwert** im Abschnitt "Spotting" auf der Registerkarte PARAMETER ein. Wenn der Spannungsschwellenwert auf Null gesetzt wird, beginnt der Verzögerungstimer sofort nach dem Start des Brenners. Wird der Spannungsschwellenwert auf über Null gesetzt, beginnt der Verzögerungszeitgeber, wenn die Lichtbogenspannung den Schwellenwert erreicht.
2. Stellen Sie die **Time On** im Abschnitt Spotting auf der PARAMETERS Tab ein. Wenn der Timer **Time On** abgelaufen ist, schaltet sich der Brenner aus. Die Zeiten sind von 0 bis 9999 Millisekunden einstellbar.

Der Brenner wird dann im G-Code mit dem Befehl **M3 $2 S1** eingeschaltet, der den Plasmabrenner als Spotting-Werkzeug auswählt.

Um den Brenner auszuschalten, verwenden Sie den G-Code-Befehl **M5 $2**.

Für weitere Informationen zu mehreren Werkzeuge siehe den gleichnamigen Abschnitt.

LinuxCNC (QtPlasmaC) erfordert eine gewisse Bewegung zwischen den Befehlen **M3** und **M5**. Aus diesem Grund ist eine minimale Bewegung bei hoher Geschwindigkeit erforderlich, um programmiert zu werden.

Ein Beispiel-G-Code ist:

```
G21 (metric)
F99999 (high feed rate)
.
.
G0 X10 Y10
M3 $2 S1 (spotting on)
G91 (relative distance mode)
G1 X0.000001
G90 (absolute distance mode)
M5 $2 (spotting off)
.
.
G0 X0 Y0
G90
M2
```

NOTE Die **hohe Vorschubgeschwindigkeit** von 99999 soll sicherstellen, dass die Bewegung bei der höchsten Vorschubgeschwindigkeit der Maschine erfolgt.

IMPORTANT EINIGE PLASMA-CUTTER SIND FÜR DIESE FUNKTION NICHT GEEIGNET.
ES WIRD EMPFOHLEN, DASS DER BENUTZER EINIGE TESTFLECKEN DURCHFÜHRT, UM SICHERZUSTELLEN, DASS DER PLASMA-CUTTER DIESE FUNKTION NUTZEN KANN.

Rohre schneiden (engl. tube cutting)

Das Rohrschneiden mit einer rotatorischen A-, B- oder C-Achse wird im G-Code-Programm wie folgt erreicht:

- #<tube_cut>=1 magischer Kommentar vor jedwedem Befehl für eine Bewegung.
- Alle Materialmessungen müssen mit den direkten Probing-Codes G38 durchgeführt werden, die unter dem Link: ../gcode/g-code.html#gcode:g38 zu finden sind.
- Alle Z-Achsenbewegungen sind erforderlich, da PlasmaC keine interne Z-Achsenbewegung während des Rohrschneidens durchführt.
- PIERCE_DELAY (engl. für Durchstichverzögerung) wird allein benötigt für die Material Parameter

- Starte einen Schnitt mit `M3 $0 S1`.
- Beende einen Schnitt mit `M5 $0`

Benutzerdefinierte Layouts für virtuelle Tastaturen

Virtuelle Tastaturunterstützung ist nur für die "integrierte" Bildschirmtastatur verfügbar. Wenn es sich noch nicht auf dem System befindet, kann es installiert werden, indem Sie Folgendes in ein Terminal eingeben:

```
sudo apt install onboard
```

Die folgenden beiden benutzerdefinierten Layouts werden für die Softkey-Unterstützung verwendet:



Zifferntastatur - wird für die Registerkarte KONVERSATION und die Registerkarte PARAMETER verwendet



Alphanumerische Tastatur - für die Bearbeitung von G-Codes und die Dateiverwaltung.

Wenn die virtuelle Tastatur neu positioniert wurde und beim nächsten Öffnen einer virtuellen Tastatur nicht sichtbar ist, klicken Sie zweimal auf das Onboard-Symbol in der Taskleiste, um die virtuelle Tastatur neu zu positionieren, so dass der Verschiebegriff sichtbar wird.

Tastatürkürzel

Nachfolgend finden Sie eine Liste aller verfügbaren Tastaturkürzel in QtPlasmaC.

NOTE Alle Tastaturkürzel sind standardmäßig deaktiviert.

Um sie nutzen zu können, müssen **KB Shortcuts** in der Sektion **GUI SETTINGS** der Registerkarte EINSTELLUNGEN aktiviert werden.

Tastatürkürzel	Action
Esc	Bricht die laufende automatische Bewegung (z. B. ein laufendes Programm, einen Messtastentest usw.) sowie einen aktiven Brennerimpuls ab (verhält sich genauso wie das Klicken auf ZYKLUS-STOPP).
F1	Schaltet den GUI-NOTAUS (engl. E-stop)-Button um (wenn der GUI-Notaus-Button aktiviert ist).
F2	Schaltet den GUI-Netzschalter um.
F9	Schaltet den Befehl "Schneiden" um, mit dem Sie einen manuellen Schnitt beginnen oder beenden können.
F12	Stylesheet-Editor anzeigen.
ALT+RETURN	Versetzt QtPlasmaC in den Modus der manuellen Dateneingabe (MDI). Beachten Sie, dass ALT + ENTER das gleiche Ergebnis erzielt. Außerdem wird das MDI-Fenster durch Drücken von RETURN (oder ENTER) geschlossen, wenn keine Eingabe in der MDI erfolgt.
`, 1-9, 0	Ändert die Jog-Geschwindigkeit auf 0%, 10%-90%, 100% des Wertes, der in der Variable DEFAULT_LINEAR_VELOCITY im **[DISPLAY]** Abschnitt der Datei <*Maschinenname*>.ini steht.
SHIFT+`, 1-9, 0	Ändert die Eilgeschwindigkeit auf 0%, 10%-90%, 100%.
CTRL+1-9, 0	Ändert die Vorschubgeschwindigkeit auf 10%-90%, 100%.
STRG+POS1	Alle Achsen werden referenziert, wenn sie noch nicht referenziert sind und in der Datei <*Maschinenname*>.ini eine Referenziersequenz eingestellt ist. Wenn sie bereits referenziert sind, werden sie nicht mehr referenziert.
CTRL+R	Zyklusstart, wenn das Programm noch nicht läuft. Zyklus fortsetzen, wenn das Programm pausiert.
ENDE	Berührt X und Y auf 0.
DEL	Ermöglicht die Verwendung eines Lasers zur Festlegung eines Ursprungs mit oder ohne Drehung. Siehe den Abschnitt LASER für detaillierte Anweisungen.
LEERTASTE (engl. space bar)	Hält die Bewegung an.
STRG+LEERTASTE	Löscht Benachrichtigungen.
O	Öffnet ein neues Programm.
L	Lädt das zuvor geöffnete Programm, wenn kein Programm geladen ist. Lädt das aktuelle Programm erneut, wenn ein Programm geladen ist.

Tastatürkürzel	Action
→	Joggt die X-Achse positiv.
←	Joggt die X-Achse negativ.
	Joggt die Y-Achse positiv.
	Joggt die Y-Achse negativ.
BILD-AUF (engl. page up)	Joggt die Z-Achse positiv.
BILD-AB (engl. page down)	Joggt die Z-Achse negativ.
[	Joggt die A-Achse positiv.
]	Joggt die A-Achse negativ.
.	Joggt die B-Achse positiv.
,	Joggt die B-Achse negativ.
SHIFT (+ Jog-Taste)	Die "Umschalttaste" (auch "Hochtaste", engl. shift key) wird zusammen mit einer beliebigen Jog-Taste verwendet, um ein schnelles Joggen auszulösen.
+ (+Jog Taste)	Die Plus-Taste kann mit jeder Jog-Taste verwendet werden, um einen schnellen Jog aufzurufen (verhält sich wie SHIFT).
- (+Jog Taste)	Die Minustaste kann mit einer beliebigen Tipptaste verwendet werden, um einen langsamen Tippbetrieb (10% der angezeigten Tippgeschwindigkeit) aufzurufen. Wenn der langsame Tippbetrieb bereits aktiv ist, wird die Achse mit der angezeigten Jog-Geschwindigkeit verfahren.

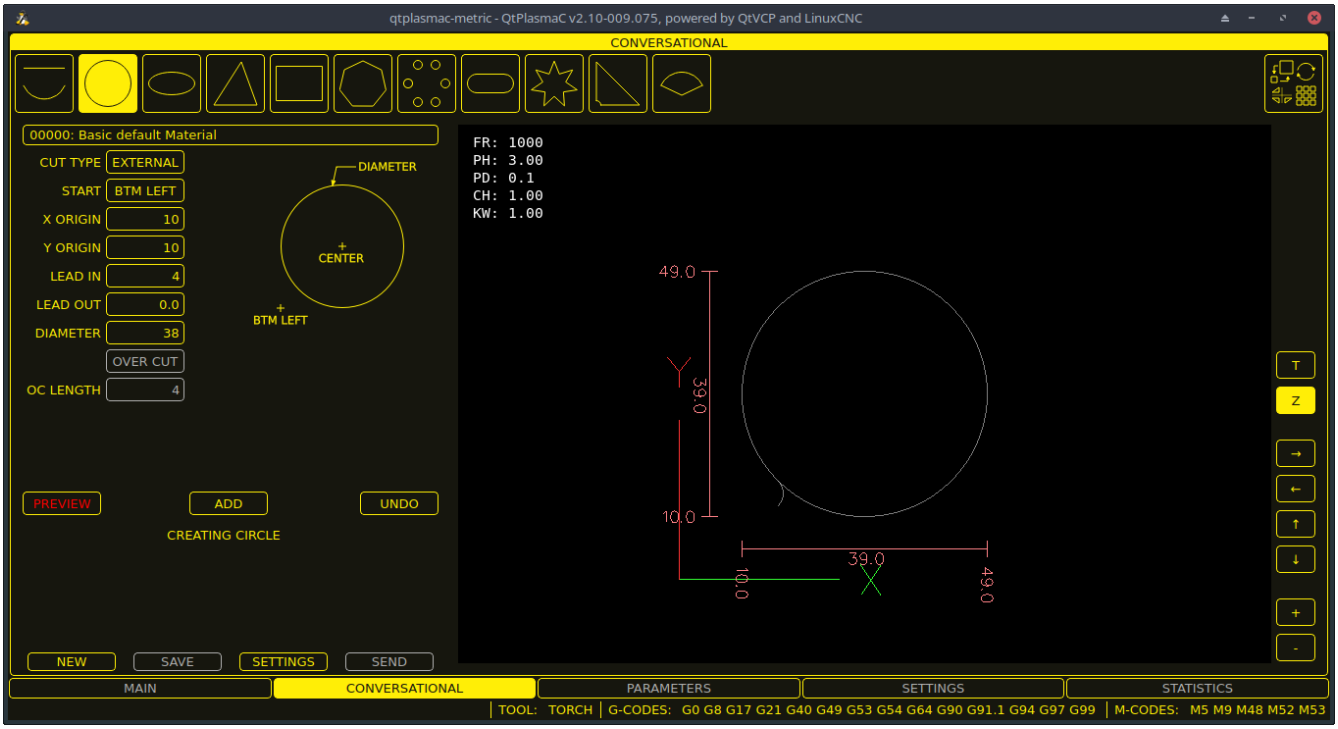
MDI

In addition to the typical G and M codes that are allowed by LinuxCNC in MDI mode, the MDI in QtPlasmaC can be used to access several other handy features. The following link outlines the features and their use: MDILine Widget

NOTE M3, M4 und M5 sind in der QtPlasmaC MDI nicht erlaubt.

Außerdem wird das MDI-Fenster geschlossen, wenn Sie RETURN (oder ENTER) drücken, ohne dass ein Eintrag in der MDI erfolgt.

10.8.10. Conversational Shape-Bibliothek



Die **Conversational Shape** (engl. für Gestalt/Form) **Bibliothek** besteht aus mehreren grundlegenden Formen und Funktionen, die den Benutzer bei der schnellen Erstellung von G-Code an der Maschine unterstützen, um einfache Formen schnell zu schneiden. Diese Funktion befindet sich auf der CONVERSATIONAL Registerkarte.

NOTE	Die Conversational Library ist nicht als CAD/CAM-Ersatz gedacht, denn es gibt Grenzen für das, was erreicht werden kann.

Bei leeren Einträgen in den Form-Eingabefeldern wird die aktuelle Einstellung zum Zeitpunkt der Erstellung des G-Codes verwendet. Wenn z.B. **X Start** leer gelassen wurde, wird die aktuelle Position der X-Achse verwendet.

Alle An- und Ableitungen (engl. lead-ins und lead-outs) sind Bögen, mit Ausnahme von **Kreisen** und **Sternen**:

Kreise (engl. circles)**:**

- Wenn der Kreis extern ist, dann ist jede Hin- oder Rückführung (lead-in or lead-out) ein Bogen.
- Wenn der Kreis innenliegend ist und ein **kleines Loch** hat, dann ist jeder Lead-in senkrecht und es gibt keinen Auslauf (engl. lead-out).
- Wenn der Kreis intern und kein **kleines Loch** ist, dann ist jeder Lead-in und Lead-out ein Bogen. Wenn der Lead-in eine Länge von mehr als der Hälfte des Radius hat, wird der Leadin senkrecht zurückgesetzt und es gibt keinen Lead-out. Wenn der Lead-out eine Länge von mehr als der Hälfte des Radius hat, gibt es keinen Lead-out.

Sterne (engl. stars)**:**

- Der Lead-in befindet sich im gleichen Winkel wie der erste Schnitt und der Leadout im gleichen Winkel wie der letzte Schnitt.

NOTE Ein **kleines Loch** ist ein Kreis, der kleiner ist als der auf der Seite KONVERSATIONSEINSTELLUNGEN (engl. conversational settings) angegebene KLEINE LOCHDURCHMESSER.

NOTE Die Löcher in einer BOLZENKREIS (engl. bolt circle)-Form werden sich ebenfalls an die obigen Regeln halten.

Die Schnittreihenfolge entspricht der Reihenfolge, in der die Form gebaut wurde.

Wenn Sie während der Bearbeitung der Parameter **Return** auf der Tastatur drücken, wird automatisch die Vorschau der Form angezeigt, wenn genügend Parameter eingegeben wurden, um die Form zu erstellen. Ein Klick auf eines der verfügbaren Kontrollkästchen bewirkt dasselbe.

Die allgemeinen Funktionen sind wie folgt:

Name	Beschreibung
Material-Dropdown	Ermöglicht es dem Benutzer, das gewünschte Material zum Schneiden auszuwählen. Wenn "MATERIAL ANZEIGEN" auf der Registerkarte EINSTELLUNGEN ausgewählt ist, wird im Konversationsvorschaufenster eine visuelle Referenz mit den wichtigsten Materialschnitteinstellungen angezeigt. Beispiele sind: Vorschubgeschwindigkeit, Einstichhöhe (engl. pierce height), Einstichverzögerung (engl. pierce delay), Schnitthöhe und Schnittfugenbreite (engl. kerf width) (nur für Konversation). Schnitt-Stromstärken (engl. cut amps) werden angezeigt, wenn die PowerMax-Kommunikation aktiviert ist.
NEU	Entfernt die aktuelle G-Code-Datei und lädt eine leere G-Code-Datei.
SPEICHERN (engl. save)	Öffnet ein Dialogfeld, in dem die aktuelle Form als G-Code-Datei gespeichert werden kann.
EINSTELLUNGEN (engl. settings)	Ermöglicht die Änderung der globalen Einstellungen.
SENDEN (engl. send)	Lädt die aktuelle Form in LinuxCNC (QtPlasmaC). Wenn die letzte Bearbeitung nicht hinzugefügt wurde, wird sie verworfen.
VORSCHAU (engl. preview)	Zeigt eine Vorschau der aktuellen Form an, sofern die erforderlichen Informationen vorhanden sind.
FORTSETZEN (engl. continue)	Diese Schaltfläche wird nur für Linien und Bögen verwendet. Ermöglicht das Hinzufügen eines weiteren Segments zum aktuellen Segment/zu den aktuellen Segmenten.
HINZUFÜGEN (engl. add)	Speichert die aktuelle Form in den aktuellen Auftrag.
RÜCKGÄNGIG MACHEN	Stellt den zuvor gespeicherten Zustand wieder her.

Name	Beschreibung
NEU LADEN	Lädt die ursprüngliche G-Code-Datei oder eine leere Datei, wenn keine geladen war.

Wenn eine G-Code Datei in LinuxCNC (QtPlasmaC) geladen ist, wenn die CONVERSATIONAL Registerkarte ausgewählt ist, wird dieser Code in die Conversational als erste Form des Jobs importiert. Wenn dieser Code nicht benötigt wird, kann er durch Drücken der Schaltfläche **NEW** entfernt werden.

Wenn eine hinzugefügte Form nicht gespeichert oder gesendet wurde, ist es nicht möglich, die Registerkarten in der grafischen Benutzeroberfläche zu wechseln. Um das Wechseln der Registerkarten wieder zu aktivieren, müssen Sie entweder die Form **SPEICHERN**, **SENDEN** oder **NEU** drücken, um die Form zu entfernen.

Wenn **NEU** gedrückt wird, um eine hinzugefügte Form zu entfernen, die nicht gespeichert oder gesendet wurde, wird ein Warndialog angezeigt.

NOTE Alle Entfernungen sind in Maschineneinheiten relativ zum aktuellen Benutzerkoordinatensystem und alle Winkel sind in Grad angegeben.

Konversationseinstellungen

Globale Einstellungen für die Formbibliothek können durch Drücken der Schaltfläche **EINSTELLUNGEN** in der CONVERSATIONAL Tab vorgenommen werden. Dadurch werden alle verfügbaren Einstellungsparameter angezeigt, die für die Erstellung von G-Code-Programmen verwendet werden. Dazu gehören:

- **Präambel**
- **Postambel**
- **Ursprung (engl. origin) (Mitte (center) oder unten links (bottom left))**
- **Einführungslänge** (engl. leadin length)
- **Ausgangslänge** (engl. leadout length)
- * Kleiner Lochdurchmesser*
- **Kleines Loch Geschwindigkeit**
- **Vorschaufenster Rastergröße**

Jeder Innenkreis mit einem Durchmesser kleiner als **Kleiner Lochdurchmesser** wird als kleine Bohrung klassifiziert und hat einen geraden Einstich (engl. lead-in) mit einer Länge, die kleiner ist als entweder der Radius der Bohrung oder die angegebene Einstichlänge. Außerdem wird die Vorschubgeschwindigkeit auf **Kleine Bohrungsgeschwindigkeit** eingestellt.

Präambel und Postambel können als eine durch Leerzeichen getrennte Folge von G-Codes und M-Codes eingegeben werden. Wenn der Benutzer möchte, dass der generierte G-Code jeden Code in einer eigenen Zeile enthält, kann er dies durch Trennen der Codes mit `\n` erreichen.

Dadurch werden alle Codes in dieselbe Zeile gesetzt:

```
G21 G40 G49 G64p0.1 G80 G90 G92.1 G94 G97
```

Dadurch wird jeder Code in eine eigene Zeile gesetzt:

```
G21\nG40\nG49\nM52P1\nG64p0.1\nG80\nG90\nG92.1\nG94\nG97
```

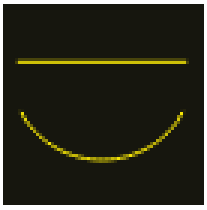
Bemerke, dass LinuxCNC nicht mehrere **P** Wörter auf derselben Zeile erlaubt.

Wenn Sie die Taste **RELOAD** drücken, werden alle geänderten, aber nicht gespeicherten Einstellungen verworfen.

Durch Drücken der Taste **SAVE** werden alle Einstellungen wie angezeigt gespeichert.

Wenn Sie die Taste **EXIT** drücken, wird das Einstellungsfeld geschlossen und Sie kehren zur vorherigen Form zurück.

Konversationslinien und Bögen



Linien und Bögen haben eine zusätzliche Option, indem sie aneinandergereiht werden können, um eine komplexe Form zu schaffen.

Es stehen zwei Linienarten und drei Bogenarten zur Verfügung:

1. **Linie** mit einem Start- und einem Endpunkt.
2. **Linie** mit einem Startpunkt, einer Länge und einem Winkel.
3. **Bogen** (engl. arc) bei gegebenem Startpunkt, Wegpunkt und Endpunkt.
4. **Arc** mit einem Startpunkt, einem Endpunkt und einem Radius.
5. **Arc** mit einem Startpunkt, einer Länge, einem Winkel und einem Radius.

Nutzung von Linien und Bögen:

1. Wählen Sie das Symbol **Linien und Bögen** aus.
2. Wählen Sie den Typ der zu erstellenden Linie oder des Bogens.
3. Wählen Sie das Material aus der Dropdown-Liste MATERIAL. Wenn kein Material ausgewählt wird, dann wird das Standardmaterial (00000) verwendet.
4. Geben Sie die gewünschten Parameter ein.
5. Drücken Sie **PREVIEW**, um die Form zu sehen.
6. Wenn Sie mit der Form zufrieden sind, drücken Sie **CONTINUE**.
7. Ändern Sie bei Bedarf den Linien- oder Bogentyp und fahren Sie mit diesem Verfahren fort, bis die

Form vollständig ist.

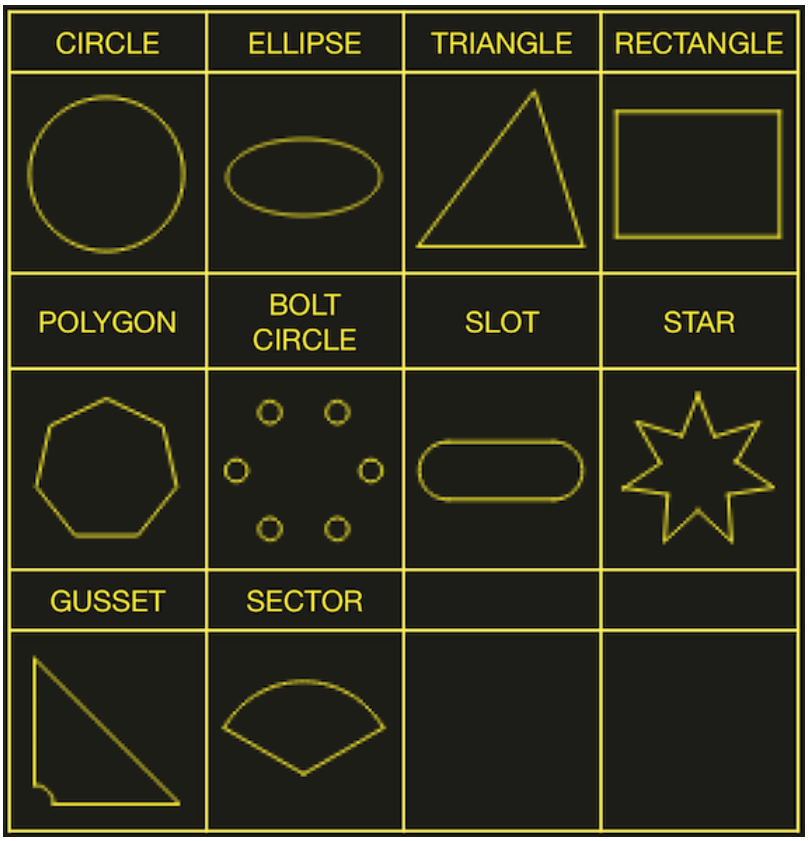
8. Drücken Sie **SEND**, um die G-Code-Datei zum Schneiden an LinuxCNC (QtPlasmaC) zu senden.

Wenn der Benutzer eine geschlossene Form erstellen möchte, muss er alle erforderlichen Anfänge als das erste Segment der Form erstellen. Wenn ein Auslauf erforderlich ist, muss dieser das letzte Segment der Form sein.

NOTE In diesem Stadium gibt es keine automatische Option für die Erstellung eines Lead-in/Lead-out, wenn die Form geschlossen ist.

Dialoggeführte Einzelform (conversational)

Die folgenden Formen sind für die Erstellung verfügbar:



So erstellen Sie eine Form:

1. Wählen Sie das entsprechende Symbol für die zu erstellende Form. Die verfügbaren Parameter werden angezeigt.
2. Wählen Sie das Material aus der Dropdown-Liste MATERIAL. Wenn kein Material ausgewählt wird, dann wird das Standardmaterial (00000) verwendet.
3. Geben Sie die entsprechenden Werte ein und drücken Sie **PREVIEW**, um die Form anzuzeigen.
4. Wenn die Form nicht korrekt ist, ändern Sie die Werte und drücken Sie **VORSCHAU** (engl. preview), damit die neue Form angezeigt wird. Wiederholen Sie den Vorgang, bis Sie mit der Form zufrieden sind.
5. Drücken Sie **ADD**, um die Form zur G-Code-Datei hinzuzufügen.

6. Drücken Sie **SEND**, um die G-Code-Datei zum Schneiden an LinuxCNC (QtPlasmaC) zu senden.

Für **KREIS** wird die Schaltfläche **ÜBERSCHNEIDEN** gültig, wenn ein SCHNITTTYP von INTERN ausgewählt ist und der in das Feld DURCHMESSER eingegebene Wert kleiner ist als der Parameter Kleiner Lochdurchmesser im Abschnitt Dialog-EINSTELLUNGEN.

Für **BOLZENKREIS** (engl. bolt circle) wird die Schaltfläche **ÜBERSCHNEIDEN** gültig, wenn der im Feld LOCHDURCHMESSER eingegebene Wert kleiner ist als der Parameter KLEINER LOCHDURCHMESSER im Abschnitt Dialog-EINSTELLUNGEN.

Bei den folgenden Formen wird KERF OFFSET aktiv, sobald ein LEAD IN angegeben wird:

1. TRIANGLE (engl. für Dreieck)
2. RECTANGLE (engl. für Rechteck)
3. POLYGON (engl. für Vieleck)
4. SLOT (engl. für Schlitz)
5. STAR (engl. für Stern)
6. GUSSET

Gruppe von Einzelformen (conversational)

Mehrere Formen können zusammengefügt werden, um eine komplexe Gruppe zu bilden.

Die Schnittreihenfolge der Gruppe wird durch die Reihenfolge bestimmt, in der die einzelnen Formen der Gruppe hinzugefügt werden.

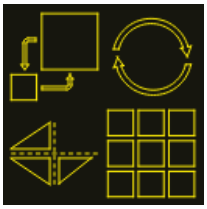
Sobald eine Form zur Gruppe hinzugefügt wurde, kann sie weder bearbeitet noch entfernt werden.

Bei Gruppen können keine Formen entfernt, sondern nur hinzugefügt werden.

Um eine Gruppe von Formen zu schaffen:

1. Erstellen Sie die erste Form wie in **Einzelne Form**.
2. Drücken Sie **ADD** und die Form wird der Gruppe hinzugefügt.
3. Wenn der Benutzer eine weitere Version der gleichen Form hinzufügen möchte, bearbeiten Sie die erforderlichen Parameter und drücken Sie **ADD**, wenn Sie mit der Form zufrieden sind.
4. Wenn der Benutzer eine andere Form hinzufügen möchte, wählen Sie diese Form aus und erstellen Sie sie wie bei einer **Einzelnen Form**.
5. Wiederholen Sie diesen Vorgang, bis alle erforderlichen Formen zur Vervollständigung der Gruppe hinzugefügt wurden.
6. Drücken Sie **SEND**, um die G-Code-Datei zum Schneiden an LinuxCNC (QtPlasmaC) zu senden.

Conversational Block



Die Funktion "Conversational Block" ermöglicht die Durchführung von Blockoperationen mit der aktuellen Form oder einer Gruppe von Formen, die im CONVERSATIONAL Registerkarte angezeigt werden. Dies kann eine G-Code-Datei einschließen, die nicht mit der Conversational Shape Library erstellt wurde, die zuvor von der Haupt-Registrierkarte geladen wurde.

Eine zuvor gespeicherte Block-G-Code-Datei kann auch über die Haupt-Registerkarte geladen und dann mit der Funktion "Conversational Block" bearbeitet werden.

Alle Blockoperationen werden in den nativ verwendeten Einheiten der Maschine ausgeführt. Wird eine Datei geöffnet, die in einem anderen Einheitensystem vorliegt, so wird die in den CONVERSATIONAL-Reiter geladene Kopie beim Aktivieren dieses Reiters automatisch in die Maschineneinheiten umgerechnet.

Blockoperationen:

- Drehen Sie
- Skala
- Array
- Mirror (engl. für Spiegel)
- Flip (engl. für umdrehen)

Um einen Block zu anzulegen:

1. Erstellen Sie eine Form oder eine Gruppe, oder verwenden Sie eine zuvor geladene G-Code-Datei.
2. Klicken Sie auf das Blocksymbol, um die Block-Tabelle zu öffnen.
3. Geben Sie die entsprechenden Werte auf der Registerkarte Block ein und drücken Sie **VORSCHAU**, um die resultierenden Änderungen anzuzeigen.
4. Wenn das Ergebnis nicht korrekt ist, ändern Sie die Werte und drücken Sie **VORSCHAU** und das neue Ergebnis wird angezeigt. Wiederholen Sie den Vorgang, bis Sie mit dem Ergebnis zufrieden sind.
5. Drücken Sie **ADD** (engl. für hinzufügen), um den Vorgang abzuschließen.
6. Drücken Sie **SEND**, um die G-Code-Datei an LinuxCNC (QtPlasmaC) zum Schneiden zu senden, oder **SAVE**, um die G-Code-Datei zu speichern.

   SPALTEN & REIHEN

   Gibt die Anzahl der in Spalten und Zeilen angeordneten Duplikate der ursprünglichen Form sowie den Abstand zwischen den Ursprüngen einer jeden Form.

URSPRUNG

Versetzt das Ergebnis von den Ursprungskoordinaten.

WINKEL

Rotation des Ergebnisses.

SCALE

Skaliert das Ergebnis.

ROTATION

Drehen der Form innerhalb des Ergebnisses.

MIRROR (engl. für Spiegel)

spiegelt die Form um ihre X-Koordinaten innerhalb des Ergebnisses.

WENDEN

spiegelt die Form um ihre Y-Koordinaten innerhalb des Ergebnisses.

Wenn das Ergebnis ein Array von Formen ist, dann ist die Schnittreihenfolge des Ergebnisses von der linken Spalte zur rechten Spalte, beginnend mit der untersten Zeile und endend mit der obersten Zeile.

"Conversational" Speichern eines Jobs

Der aktuelle Job, der im Preview Panel angezeigt wird, kann jederzeit mit dem unteren **SAVE** Button gespeichert werden. Wenn der G-Code an LinuxCNC (QtPlasmaC) gesendet wurde und der Benutzer die CONVERSATIONAL Registerkarte verlassen hat, kann der Benutzer die G-Code Datei immer noch von der GUI aus speichern. Alternativ kann der Benutzer auf den CONVERSATIONAL Registerkarte klicken, wodurch der Job neu geladen wird, woraufhin er die Schaltfläche **SAVE** drücken kann.

10.8.11. Fehlermeldungen

Fehlerprotokollierung

Alle Fehler werden im Maschinenlog protokolliert, das in der Registerkarte zu STATISTIKEN eingesehen werden kann. Die Logdatei wird in das Konfigurationsverzeichnis gespeichert, wenn QtPlasmaC heruntergefahren wird. Die letzten fünf Logfiles werden aufbewahrt, danach wird das älteste Logfile bei jeder Erstellung eines neuen Logfiles gelöscht. Diese gespeicherten Logdateien können mit einem beliebigen Texteditor eingesehen werden.

Anzeige von Fehlermeldungen

Standardmäßig zeigt QtPlasmaC Fehlermeldungen über ein Popup-Fenster "Operator Error" an. Darüber hinaus weist QtPlasmaC den Benutzer darauf hin, dass ein Fehler an das Maschinenprotokoll gesendet wurde, indem die Meldung **"ERROR SENT TO MACHINE LOG "** im unteren linken Teil der Statusleiste angezeigt wird.

Der Benutzer kann das Popup-Fenster für Bedienerfehler deaktivieren und die Fehlermeldungen auf der

Registerkarte STATISTICS anzeigen, indem er die folgende Option in der Datei <*Maschinenname*>.prefs im Verzeichnis <*Maschinenname*> unter **[SCREEN_OPTIONS]** (engl. für Bildschirm-Optionen) auf **False** ändert:

```
desktop_notify
```

NOTE <*Maschinenname*>.prefs muss bei geschlossenem QtPlasmaC bearbeitet werden, sonst werden alle Änderungen beim Beenden überschrieben.

Zusätzlich ist es möglich, **ERROR SENT TO MACHINE LOG** blinken zu lassen, um die Aufmerksamkeit des Benutzers zu erregen, indem man die folgende Option in den **[GUI_OPTIONS]** Abschnitt der Datei <*machine_name*>.prefs einfügt oder bearbeitet:

```
Flash error = True
```

Kritischer Fehler

Es gibt eine Reihe von Fehlermeldungen, die von QtPlasmaC ausgegeben werden, um den Benutzer über auftretende Fehler zu informieren. Die Meldungen können in zwei Gruppen unterteilt werden: **Kritisch** und **Warnung**.

Kritische Fehler (engl. critical errors) führen dazu, dass das laufende Programm angehalten wird, und der Bediener muss die Fehlerursache beseitigen, bevor er fortfahren kann.

Wenn der Fehler während des Schneidens aufgetreten ist, so ist eine Vorwärts- oder Rückwärtsbewegung erlaubt, während die Maschine angehalten wird, damit der Benutzer die Maschine neu positionieren kann, bevor sie den Schnitt wieder aufnimmt.

Wenn der Fehler behoben ist, kann das Programm fortgesetzt werden.

Diese Fehler zeigen an, dass der entsprechende Sensor während des Schneidens aktiviert wurde:

- **Abreißschalter aktiviert, Programm wird angehalten**
- **Schwimmerschalter aktiviert, Programm wird angehalten**
- **aktiviertes Ohmic Probe Programm ist angehalten**

Diese Fehler deuten darauf hin, dass der entsprechende Sensor aktiviert wurde, bevor die Sondierung begann:

- **ohmische Sonde erkannt, bevor das Sondenprogramm pausiert wird**
- **Schwimmerschalter erkannt, bevor das Antastprogramm pausiert wird**
- **Ausbruchsschalter erkannt, bevor das Antastprogramm unterbrochen wird**

Das Lichtbogen-OK-Signal ging während der Schneidbewegung verloren, bevor der Befehl **M5** erreicht wurde:

- **gültiger Lichtbogen verloren Programm wird angehalten**

Die Z-Achse erreichte die untere Grenze, bevor das Werkstück erkannt wurde:

- **Untere Grenze erreicht, während das Sondierungsprogramm pausiert wird**

Das Werkstück ist zu hoch, um es sicher und schnell entfernen zu können:

- **Material zu hoch für sicheres Verfahren, Programm wird angehalten**

Einer dieser Werte im Abschnitt MATERIAL der Registerkarte PARAMETER ist ungültig (z.B. wenn sie auf Null gesetzt sind):

- **Ungültige Lochstechhöhe (engl. pierce height) oder ungültige Schnitthöhe oder ungültige Schnittspannung, Programm ist pausiert**

Es wurde kein Lichtbogen erkannt, nachdem versucht wurde, so oft zu starten, wie in **Max Starts** im ARC-Rahmen des CONFIGURATION-Abschnitts der PARAMETERS Tab angegeben:

- **kein Lichtbogen erkannt nach <*n*>d Startversuchen Programm ist pausiert**
- **kein Lichtbogen erkannt nach <*n*>d Startversuchen Manueller Schnitt wird gestoppt**

THC hat dazu geführt, dass die Untergrenze beim Schneiden erreicht wurde:

- **Untere Grenze erreicht, während das THC-Abwärtsprogramm pausiert**

THC hat dazu geführt, dass die Obergrenze beim Schneiden erreicht wird:

- **Höchstgrenze erreicht, während das THC-Aufstiegsprogramm pausiert ist**

Diese Fehler deuten darauf hin, dass die Höhe der Bewegung zum Durchstechen den MAX_LIMIT-Wert der Z-Achse für die entsprechende Messmethode überschreiten würde:

- **Die Höhe der Sonde würde den maximalen Grenzwert der Z-Achse überschreiten, der beim Anfahren der Sondenhöhe während der Abtastung des Schwimmerschalters festgestellt wurde**
- **Durchstech-Höhe würde den maximalen Grenzwert der Z-Achse überschreiten, der beim Anfahren der Sondenhöhe während der ohmschen Abtastung festgestellt wurde**

Diese Fehler deuten darauf hin, dass die Höhe der Bewegung zum Lochstechen die maximale sichere Höhe der Z-Achse für die entsprechende Sondenmethode überschreiten würde:

- **Einstechhöhe würde die maximale sichere Höhe der Z-Achse überschreiten, die beim Abtasten des Schwimmerschalters gefunden wurde**
- **Einstechhöhe würde die maximale sichere Höhe der Z-Achse überschreiten, die beim Abtasten des Schwimmerschalters gefunden wurde**

Warnhinweise

Warnmeldungen halten ein laufendes Programm nicht an und haben nur informativen Charakter.

Diese Meldungen zeigen an, dass der entsprechende Sensor aktiviert wurde, bevor ein Sondentest begann:

- **ohmsche Sonde erkannt, bevor Sondentest abgebrochen wurde**
- **Schwimmerschalter erkannt, bevor Sondentest abgebrochen wurde**
- **Abreißschalter erkannt, bevor der Sondentest abgebrochen wurde**

Dies zeigt an, dass der entsprechende Sensor während eines Verbrauchsmaterialwechsels aktiviert wurde:

- **Abrissschalter, Schwimmer oder ohmscher Widerstand während des Verbrauchsmaterialwechsels aktiviert, Bewegung wurde angehalten**
  WARNUNG: DIE BEWEGUNG WIRD SOFORT WIEDER AUFGENOMMEN, WENN DIESER ZUSTAND BEHOBEN IST!

WARNING	DIE BEWEGUNG BEIM WECHSEL DES VERBRAUCHSMATERIALS WIRD SOFORT WIEDER AUFGENOMMEN, SOBALD DIE ENTSPRECHENDE SENSORAKTIVIERUNG BEHOBEN IST.

Dies zeigt an, dass der entsprechende Sensor während der Sondenprüfung aktiviert wurde:

- **Abreißschalter während des Sondentests entdeckt**

Dies deutet darauf hin, dass der Tastkopfkontakt verloren ging, bevor der Nullpunkt gefunden wurde:

- **Fehler bei der Sondenauslösung während der Sondenprüfung**

Dies zeigt an, dass die untere Grenze während eines Sondentests erreicht wurde:

- **Unterer Grenzwert bei Sondentests erreicht**

Dies zeigt an, dass das Anfahren der Lochstechhöhe bei der entsprechenden Antastmethode den MAX_LIMIT-Wert der Z-Achse überschreiten würde:

- **Durchstech-Höhe würde den maximalen Grenzwert der Z-Achse überschreiten, der beim Anfahren der Sondenhöhe während der Prüfung der Schwimmerschalter-Sonde festgestellt wurde**
- **Durchstech-Höhe würde den maximalen Grenzwert der Z-Achse überschreiten, der beim Anfahren der Sondenhöhe während der ohmschen Sondenprüfung festgestellt wurde**

Dies zeigt an, dass die sichere Höhe reduziert wurde, weil THC die Z-Achse während des Schneidens anhebt:

- **sichere Verfahrhöhe wurde reduziert**

Dies zeigt an, dass der Wert für die Lichtbogenspannung ungültig war (NAN oder INF), als QtPlasmaC gestartet wurde.

- **invalid arc-voltage-in**

10.8.12. Aktualisierung von QtPlasmaC

Standard-Update

QtPlasmaC-Update-Hinweise werden veröffentlicht unter https://forum.linuxcnc.org/plasmac/37233-plasmac-updates.

Benutzern wird dringend empfohlen, einen Benutzernamen zu erstellen und den obigen Thread zu abonnieren, um Benachrichtigungen über Aktualisierungen zu erhalten.

Bei einer Standard-ISO-Installation wird LinuxCNC nur aktualisiert, wenn ein neues Minor-Release veröffentlicht wurde. QtPlasmaC wird dann automatisch seine Konfiguration aktualisieren, wenn es das erste Mal nach einem LinuxCNC-Update ausgeführt wird.

LinuxCNC wird normalerweise durch die Eingabe der folgenden Befehle in ein Terminalfenster (einer nach dem anderen) aktualisiert:

```
sudo apt update
sudo apt dist-upgrade
```

Kontinuierliche Aktualisierung

Verbesserungen und Fehlerkorrekturen werden nicht auf einer Standardinstallation verfügbar sein, bis eine neue kleinere Version von LinuxCNC veröffentlicht wurde. Wenn der Benutzer aktualisieren möchte, wenn eine neue QtPlasmaC Version veröffentlicht wurde, kann er das LinuxCNC Buildbot Repository anstelle des Standard LinuxCNC Repository verwenden, indem er den Anweisungen folgt bei http://buildbot.linuxcnc.org/ .

10.8.13. Ändern einer bestehenden QtPlasmaC Konfiguration

Es gibt zwei Möglichkeiten, eine bestehende QtPlasmaC-Konfiguration zu ändern:

1. Ausführen des entsprechenden << configuring,Konfigurationsassistenten>> und Laden der vom Assistenten gespeicherten .conf-Datei.
2. Manuelles Bearbeiten der INI und/oder der HAL-Datei der Konfiguration.

IMPORTANT	Jede manuelle Änderung an den Dateien <*Maschinenname*>.ini und <*Maschinenname*>.hal wird nicht in PnCConf oder StepConf registriert.

NOTE	Wenn Sie sich nicht sicher sind, wie der vollständige Name des HAL-Pins lautet, können Sie LinuxCNC starten und **HalShow** ausführen, um eine vollständige Liste aller HAL-Pins zu erhalten.

10.8.14. QtPlasmaC GUI anpassen

Das Styling der QtPlasmaC-GUI erfolgt mit Qt-Stylesheets und einige Anpassungen können durch die Verwendung eines eigenen Stylesheets erreicht werden. Dies ermöglicht dem Benutzer, einige GUI-

Elemente wie Farbe, Rahmen, Größe usw. zu ändern. Das Layout der GUI kann damit nicht verändert werden.

Informationen zu Qt-Stylesheets sind hier verfügbar.

Es gibt zwei Methoden, um benutzerdefinierte Stile anzuwenden:

1. Einen benutzerdefinierten Stil hinzufügen: Verwenden Sie diese Option für kleinere Stiländerungen.
2. Einen neuen Stil erstellen Verwenden Sie diese Option für eine vollständige Stiländerung.

Einen benutzerdefinierten Stil hinzufügen

Das Hinzufügen von Stiländerungen zum Standard-Stylesheet erfolgt durch Erstellen einer Datei im Konfigurationsverzeichnis <*Maschinenname*>. Diese Datei MUSS den Namen qtplasmac_custom.qss tragen. Alle erforderlichen Stiländerungen werden dann zu dieser Datei hinzugefügt.

Beispielsweise könnte der Benutzer eine rote Anzeige der Lichtbogenspannung, eine größere grüne LED für das Einschalten des Brenners und eine größere Taste zum Aktivieren des Brenners wünschen. Dies würde mit folgendem Code in qtplasmac_custom.qss erreicht:

```
#arc_voltage {
    color: #ff0000 }

#led_torch_on {
    qproperty-diameter: 30;
    qproperty-color: green }

#torch_enable::indicator {
    width: 30;
    height: 30}
```

Erstellen eines neuen Stils

Benutzerdefinierte Stylesheets werden durch eine der folgenden Einstellungen im Abschnitt **[GUI_OPTIONS]** der Datei <*Maschinenname*>.prefs aktiviert. Diese Option muss gesetzt werden auf den Dateinamen des Stylesheets wie unten gezeigt.

```
Custom style = the_cool_style.qss
```

Der Dateiname kann ein beliebiger gültiger Dateiname sein. Die Standarderweiterung lautet .qss, ist aber nicht zwingend erforderlich.

Es gibt einige Einschränkungen für das benutzerdefinierte Stylesheet für QtPlasmaC, z.B. sind die Jog-Buttons, Cut-Recovery-Buttons und die Conversational Shape-Buttons Bilddateien und können nicht benutzerdefiniert gestaltet werden.

Die benutzerdefinierte Stildatei benötigt eine Kopfzeile im folgenden Format:

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

```
Benutzerdefiniertes Stylesheet Kopfzeile

color1 = #000000
#QtPlasmaC default = #ffee06

color2 = #e0e0e0
#QtPlasmaC default = #16160e

color3 = #c0c0c0
#QtPlasmaC default = #ffee06

color4 = #e0e0e0
#QtPlasmaC default = #26261e

color5 = #808080
#QtPlasmaC default = #b0b0b0

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

Die Farben können in jedem gültigen Stylesheet-Format angegeben werden.

Die oben genannten Farben werden für die folgenden Widgets verwendet. Daher muss jedes benutzerdefinierte Styling diese berücksichtigen. Die unten gezeigten Farben sind die Standardwerte, die in QtPlasmaC zusammen mit dem Farbnamen aus der Registerkarte EINSTELLUNGEN, verwendet werden.

Farbe	Parameter	Auswirkungen
color1 (#ffee06)	Vordergrund	foreground of jog buttons foreground of latching user buttons foreground of camera/laser buttons foreground of conversational shape buttons background of active conversational shape buttons
color2 (#16160e)	Hintergrund	background of latching user buttons background of camera/laser buttons background of G-code editor active line background of conversational shape buttons
color3 (#ffee06)	Hervorhebung	background of active latching user buttons background of active camera/laser buttons foreground of G-code editor cursor
color4 (#36362e)	Alternativer Hintergrund	Hintergrund der aktiven Zeile der G-Code-Anzeige
color5 (#b0b0b0)	Deaktiviert	Vordergrund der deaktivierten Buttons

Rückkehr zum Standardstil

Der Benutzer kann jederzeit zum Standard-Styling zurückkehren, indem er die folgenden Schritte

ausführt:

1. Schließen von QtPlasmaC, falls geöffnet.
2. Löschen Sie qtplasmac.qss aus dem Maschinen-Konfigurationsverzeichnis.
3. Löschen von qtplasmac_custom.qss aus dem Maschinen-Konfigurationsverzeichnis (falls vorhanden).
4. Öffnen Sie die Datei <*Maschinenname*>.prefs.
5. Löschen Sie den Abschnitt **[COLOR_OPTIONS]**.
6. Löschen der Option Benutzerdefinierter Stil (engl. custom style) aus dem Abschnitt **[GUI_OPTIONS]**.
7. Speichern Sie die Datei.

Beim nächsten Laden von QtPlasmaC wird das gesamte benutzerdefinierte Styling entfernt und das Standard-Styling wird wiederhergestellt.

Nachfolgend finden Sie ein Beispiel für den Abschnitt und die Optionen, die aus der <*Maschinenname*>.prefs Datei zu löschen sind:

```
[COLOR_OPTIONS]
Foreground = #ffee06
Highlight = #ffee06
LED = #ffee06
Background = #16160e
Background Alt = #36362e
Frames = #ffee06
Estop = #ff0000
Disabled = #b0b0b0
Preview = #000000
```

Benutzerd-angepasster Python-Code

Es ist möglich, benutzerdefinierten Python-Code hinzuzufügen, um einige bestehende Funktionen zu ändern oder neue hinzuzufügen. Benutzerdefinierter Code kann auf zwei verschiedene Arten hinzugefügt werden: über eine Benutzerbefehlsdatei oder eine periodische Benutzerdatei.

Eine Benutzerbefehlsdatei wird im Abschnitt DISPLAY der Datei <*Maschinenname*>.ini angegeben und enthält Python-Code, der während des Starts verarbeitet wird.

```
USER_COMMAND_FILE = my_custom_code.py
```

Eine periodische Benutzerdatei muss den Namen user_periodic.py tragen und im Konfigurationsverzeichnis des Rechners abgelegt werden. Diese Datei wird in jedem Zyklus (in der Regel 100 ms) verarbeitet und wird für Funktionen verwendet, die regelmäßig aktualisiert werden müssen.

Benutzerdefinierte G-Code-Filter

Aller eingehender G-Code wird von einem G-Code-Filter geparst (gelesen und auf Korrektheit überprüft), um sicherzustellen, dass es für QtPlasmaC geeignet ist. Es ist möglich, diesen Filter mit

benutzerdefiniertem Python-Code aus einer im Konfigurationsverzeichnis ausgeführten Datei zu erweitern, um verschiedene Geschmacksrichtungen von G-Code in ein für QtPlasmaC geeignetes Format zu konvertieren.

Der Name dieser Datei ist custom_filter.py und wird automatisch verwendet, wenn sie existiert.

Es sind drei voreingestellte Methoden verfügbar:

Name	Funktion
custom_pre_process	Dies führt zu einer grundsätzlichen Bearbeitung jeder Zeile, bevor eine Verarbeitung im Filter erfolgt.
custom_pre_parse	Dies parst jeden G-Code aus einer Zeile **vor** irgendwelchem im Filter ausgeführte Parsen.
custom_post_parse	Dies parst jeden G-Code aus einer Zeile **nach** irgendwelchem im Filter ausgeführte Parsen.

Diese Methoden werden nach folgendem Verfahren angewandt:

- Definieren Sie die Methode mit einem Argument für die eingehenden Daten.
- Fügen Sie einen beliebigen Code hinzu, um die Daten zu manipulieren.
- Rückgabe der resultierenden Daten.
- Fürgen Sie die neue Methode hinzu.

Ein Beispiel um jeden Code zu entfernen der mit *G71* beginnt und *M2* auf *M5 $0* und *M2* zu ändern:

```
def custom_pre_parse(data):
    if data[:3] == 'G71':
        return(None)
    if data == 'M2':
        return(f'M5 $0\n\n{data}')
    return(data)
self.custom_pre_parse = custom_pre_parse
```

Darüber hinaus ist es auch möglich, jedes vorhandene Verfahren im Filter gleich zu ein anderes zu ersetzen. Dies erfordert die Definition der gleichen Anzahl von Argumenten wie die bestehende Methode, zu bemerken ist, dass *self* im Original kein Argument darstellt.

```
def new_method_name(data):
    if data[:3] == 'G71':
        return(None)
    return(data)
self.old_method_name = new_method_name
```

NOTE Der vorhandene Filtercode kann im Datei /bin/qtplasmac_gcode beobachtet werden. Die Datei sim/qtplasmac/custom_filter.py hat Beispiel Skelett-Code für benutzerdefinierte Filterung.

10.8.15. QtPlasmaC Fortgeschrittene Themen

Benutzerdefinierte Buttons

Die QtPlasmaC-GUI bietet Benutzerschaltflächen, die durch Hinzufügen von Befehlen im Abschnitt USER BUTTON ENTRIES der Registerkarte EINSTELLUNGEN in der Datei <*Machinenname*>.prefs angepasst werden können.

Die Anzahl der Benutzertasten variiert je nach Anzeigetyp und Auflösung wie folgt:

- 16:9 und 4:3 - Minimum 8, Maximum 20
- 9:16 - Minimum 15, Maximum 20

Der Benutzer muss QtPlasmaC bei der gewünschten Bildschirmgröße ausführen, um festzustellen, wie viele Benutzertasten zur Verfügung stehen.

Alle Einstellungen der Datei <*Maschinenname*>.prefs für die Tasten befinden sich im Abschnitt **[BUTTONS]**.

Button-Namen

Der Text, der auf einem Button erscheint, wird auf folgende Weise festgelegt:

```
n Name = HAL Show
```

Dabei steht *n* für die Nummer des Button und **HAL Show** für den Text.

Bei Text in mehreren Zeilen trennen Sie den Text mit einem \ (Backslash):

```
n Name = HAL\Show
```

Wenn ein Ampersand als Text angezeigt werden soll, sind zwei aufeinander folgende Ampersands erforderlich:

```
n Name = PIERCE&&CUT
```

Button Code

Die Schaltflächen können folgende Funktionen ausführen:

1. Externe Befehle
2. External Python Skripte
3. G-code Befehle
4. Dual code
5. Umschalten eines HAL-Pins
6. Umschalten des Ausrichtungslasers HAL-Pin
7. einen HAL-Pin pulsieren

8. Sondentest
9. ohmscher Test
10. Schnittart
11. Verbrauchsmaterialien wechseln
12. Ein G-Code-Programm laden
13. Schalte die Brenner ein
14. Einzelner unidirektionaler Schnitt
15. Einen Auftrag einrahmen (engl. framing a job)
16. Manuellen Schnitt beginnen/beenden
17. Anzeige/Verstecken eines Offsets-Viewers
18. Laden Sie die letzte geänderte NGC-Datei, die in einem Verzeichnis gefunden wurde
19. Anzeige/Verstecken des Online-HTML-Benutzerhandbuchs
20. Umschalten zwischen Gelenk- und Teleop-Modus

Externe Befehle

Um einen externen Befehl auszuführen, wird dem Befehl ein %-Zeichen vorangestellt.

```
Code = %halshow
```

Externe Python-Skripte

Um ein externes Python-Skript auszuführen, muss dem Skriptnamen ein %-Zeichen vorangestellt werden, und es benötigt außerdem die Erweiterung .py. Es ist zulässig, das Zeichen ~ als Verknüpfung für das Heimatverzeichnis des Benutzers zu verwenden.

```
Code = %~/user_script.py
```

G-Code

Um G-Code auszuführen, geben Sie einfach den Code ein, der ausgeführt werden soll.

```
Code = G0 X100
```

Um ein vorhandenes Unterprogramm auszuführen.

```
Code = o<the_subroutine> call
```

Variablen in der Datei <*Maschinenname*>.ini können mit dem Standard LinuxCNC G-Code Format eingegeben werden. Wenn Ausdrücke enthalten sind, müssen diese in eckige Klammern gesetzt werden.

```
Code = G0 X#<_ini[joint_0]home> Y1
Code = G53 G0 Z[#<_ini[axis_z]max_limit> - 1.001]
```

Variablen in der Datei <*Maschinenname*>.prefs und auch in <*Machinenname*>.ini können eingegeben werden, indem jede Option in **{}** gesetzt wird. Sie müssen ein Leerzeichen nach jedem **{}** setzen, wenn eines der folgende Zeichen auftritt. Wenn Ausdrücke enthalten sind, müssen diese in eckige Klammern gesetzt werden.

```
BUTTON_n_CODE = G0 X{LASER_OFFSET X axis} Y{LASER_OFFSET Y axis}
BUTTON_n_CODE = G0 X{JOINT_0 HOME} Y1
BUTTON_n_CODE = G53 G0 Z[{AXIS_Z MAX_LIMIT} - 1.001]
```

Mehrere Codes können ausgeführt werden, indem die Codes mit einem "\" (Backslash) getrennt werden. Eine Ausnahme bilden die Spezialbefehle, für die ein einziger Befehl pro Taste erforderlich ist.

```
n Code = G0 X0 Y0 \ G1 X5 \ G1 Y5
```

Externe Befehle und G-Code können durch derselben Button gemischt werden.

```
n Code = %halshow \ g0x.5y.5 \ %halmeter
```

Dualer Code

Dual Code ermöglicht den Betrieb von zwei Code-Schnipseln abwechselnd bei jedem Tastendruck. Der Schaltflächentext wird auch mit jedem Tastendruck wechseln und das Anzeigelicht kann optional aktiviert werden.

Es ist zwingend erforderlich, die Button-Kodierung in der folgenden Reihenfolge anzugeben: "Dual-Code", den ersten Code, den alternativen Buttontext und den zweiten Code, jeweils durch Doppel-Semicolons getrennt. Wenn ein Indikator benötigt wird, so füge optional hinzu ";; true" am Ende.

```
n Code = dual-code ;; code1 ;; name1 ;; code2 ;; true
```

Beim ersten Drücken der Schaltfläche, wird Code1 ausgeführt, der Schaltflächentext wird auf den Namen 1 geändert, und wenn "wahr" angegeben wird, wird die Anzeige erhellt.

Beim zweiten Drücken wird Code2 ausgeführt, der Buttontext wird auf Name n geändert, und Anzeige wird wieder dunkel, sofern zuvor beleuchtet.

`code1` und `code2` folgen beide den Regeln der vorangegangenen Code-Erklärungen, Externe Befehle, Python-Code und G-code. Mehrere Codes sowie eine Mischung unterschiedlicher Codes sind zulässig.

Der folgende Code ermöglicht es dem Benutzer, eine einzelne Schaltfläche zu verwenden, um zwei Code-Snippets abwechselnd bei jedem Tastendruck auszuführen:

```
n Name = X+10
n Code = dual-code ;; G91\G0X10\G90 ;; X-10 ;; G91\G0X-10\G90
```

Dessen ursprüngliche Beschriftung lautet X+10. Wenn er gedrückt wird, bewegt sich die Brennerposition um 10 in der positiven X-Achse, und die Beschriftung ändert sich zu X-10. Wird er erneut gedrückt, bewegt sich der Brenner um 10 in der negativen X-Achse, und die Beschriftung ändert sich wieder zu X+10.

Spezielle Befehle

Die folgenden Befehle müssen jeweils als einzelner Befehl pro Benutzerbutton ausgeführt werden, und der Button-Code muss mit dem speziellen Befehl beginnen. Eine Ausnahme bildet toggle-laser, der an beliebiger Stelle im Code erscheinen darf, wie unten gezeigt.

HAL Pin umschalten

Der folgende Code ermöglicht es dem Benutzer, den aktuellen Zustand eines HAL-Bit-Pins über eine Schaltfläche zu invertieren:

```
Code = toggle-halpin the-hal-pin-name
```

Dieser Code muss als einzelner Befehl verwendet werden und darf nur einen HAL-Bit-Pin pro Taste steuern.

Die Farben der Tasten richten sich nach dem Zustand des HAL-Pins.

Nach dem Einstellen des Codes werden beim Anklicken der Schaltfläche die Farben invertiert und der HAL-Pin wechselt den Pin-Status. Die Schaltfläche bleibt "verriegelt", bis die Schaltfläche erneut angeklickt wird, wodurch die Schaltfläche wieder die ursprünglichen Farben und der HAL-Pin den ursprünglichen Pin-Status annimmt.

Es ist auch möglich, dass der Benutzer einen alternativen Text angibt, der auf der Schaltfläche angezeigt wird, während er je im eingerasteten Zustand ist. Um den alternativen Text anzugeben, verwenden Sie ein Doppelsemikolon, gefolgt von dem gewünschten Text. Dies muss der letzte Element im Button-Code sein.

```
Code = toggle-halpin the-hal-pin-name ;; PIN\TOGGLED
```

Es gibt drei externe HAL-Pins, die als Ausgang umgeschaltet werden können, die Pin-Namen sind `qtplasmac.ext_out_0`, `qtplasmac.ext_out_1` und `qtplasmac.ext_out_2`. HAL-Verbindungen zu diesen HAL-Pins müssen in einer Postgui-HAL-Datei angegeben werden, da die HAL-Pins nicht verfügbar sind, bis die QtPlasmaC-GUI geladen ist.

Bei Toggle-Halpin-Buttons kann der Benutzer den zugehörigen HAL-Pin als "EIN" markieren, bevor ein Schnittzyklus gestartet wird, indem er "cutcritical" (engl. für kritisch für Schnitt) nach dem HAL-Pin in den Tastencode einfügt. Wenn **TORCH ENABLE** (engl. für Brenner aktivieren) aktiviert ist und **CYCLE START** (engl. für Zyklusstart), **MANUAL_CUT** (engl. für manueller Schnitt), oder **SINGLE_CUT** (engl. für einzelner Schnitt) initiiert wird während der "cutcritical"-Button nicht "ON" ist, erhält der Benutzer einen Dialog, der ihn darauf hinweist und ihn auffordert, fortzufahren oder abzubrechen. Der Dialog listet alle nicht getoggelten ("umgeschalteten") Buttons mit einem entsprechenden Kontrollkästchen auf und ermöglicht es dem Benutzer auszuwählen, welche Buttons beim Klicken auf CONTINUE automatisch getoggelt werden sollen.

```
Code = toggle-halpin the-hal-pin-name cutcritical
```

Ausrichtungslaser HAL Pin umschalten

Mit dem folgenden Code kann der Benutzer den aktuellen Zustand des Ausrichtungslaser-HAL-Bitstifts

über eine Schaltfläche umkehren:

```
Code = toggle-laser
```

Dieser Code kann auch als Mehrfachbefehl mit G-Code oder externen Befehlen verwendet werden, darf aber nur den Ausrichtungslaser-HAL-Bit-Pin steuern.

Die Farben der Schaltflächen richten sich nach dem Zustand des Ausrichtungslaser-HAL-Pins.

Nach dem Einstellen des Codes werden beim Anklicken der Schaltfläche die Farben invertiert und der HAL-Pin des Ausrichtungslasers wechselt den Pin-Status. Die Schaltfläche bleibt "verriegelt", bis die Schaltfläche erneut angeklickt wird, wodurch die Schaltfläche wieder die ursprünglichen Farben und der Ausrichtungslaser-HAL-Pin den ursprünglichen Pin-Status annimmt.

Der folgende Code würde es dem Benutzer ermöglichen, den aktuellen Zustand des Ausrichtungslasers HAL-Bit-Pin über eine Schaltfläche zu invertieren und dann die X- und Y-Achsen auf den in der Datei <*Maschinen-Name*>.prefs angegebenen Offset für den Ausrichtungslaser zu bewegen:

```
Code = G0 X{LASER_OFFSET X axis} Y{LASER_OFFSET Y axis} \ toggle-laser
```

Die Position des Befehls "Toggle-Laser" ist nicht wichtig, da er unabhängig von der Position immer als erster Befehl ausgeführt wird.

Impuls HAL-Pin

Mit dem folgenden Code kann der Benutzer eine Taste verwenden, um einen HAL-Bit-Pin für eine Dauer von 0,5 Sekunden zu pulsieren:

```
Code = pulse-halpin the-hal-pin-name 0.5
```

Dieser Code muss als einzelner Befehl verwendet werden und darf nur einen HAL-Bit-Pin pro Taste steuern.

Die Impulsdauer wird in Sekunden angegeben. Wird die Impulsdauer nicht angegeben, wird sie standardmäßig auf eine Sekunde festgelegt.

Die Farben der Tasten richten sich nach dem Zustand des HAL-Pins.

Nach dem Einstellen des Codes werden beim Klicken auf die Schaltfläche die Farben der Schaltfläche und der Zustand des HAL-Pins invertiert, und die verbleibende Zeit wird auf der Schaltfläche angezeigt. Die Farbe der Schaltfläche und der Status des Pins bleiben invertiert, bis der Timer für die Impulsdauer abgelaufen ist. Danach erhält die Schaltfläche wieder ihre ursprünglichen Farben, der HAL-Pin seinen ursprünglichen Pin-Status und den ursprünglichen Namen der Schaltfläche.

Ein aktiver Impuls kann durch erneutes Klicken auf die Schaltfläche abgebrochen werden.

Es gibt drei External HAL Pins, die als Ausgang gepulst werden können, die Pin-Namen sind qtplasmac.ext_out_0, qtplasmac.ext_out_1 und qtplasmac.ext_out_2. HAL-Verbindungen zu diesen HAL-Pins müssen in einer Postgui-HAL-Datei angegeben werden, da die HAL-Pins erst verfügbar sind,

nachdem die QtPlasmaC-GUI geladen wurde.

Sonden-Test

QtPlasmaC startet eine Sonde und wenn das Material erkannt wird, steigt die Z-Achse auf die Pierce-Höhe, die derzeit im Abschnitt MATERIAL der PARAMETER Registerkarte angezeigt wird. Wenn der Benutzer im Abschnitt GUI-SETTINGS der EINSTELLUNGEN (engl. settings) Registerkarte die Option "Material anzeigen" (engl. view material) ausgewählt hat, wird dieser Wert in der oberen linken Ecke des PREVIEW-Fensters neben **PH:** angezeigt.

QtPlasmaC wartet dann in diesem Zustand für die angegebene Zeit (ohne Nachkommastellen gerundet), bevor die Z-Achse in die Ausgangsposition zurückkehrt. Ein Beispiel für eine Verzögerung von 6 Sekunden finden Sie unten. Wird keine Zeit angegeben, so wird die Probezeit standardmäßig auf 10 Sekunden gesetzt.

```
Code = probe-test 6
```

NOTE Durch Aktivieren einer Benutzerschaltfläche als Sondentest-Schaltfläche wird ein externer HAL-Pin hinzugefügt, der von einem Anhänger (engl. pendant) usw. verbunden werden kann. HAL-Verbindungen zu diesem HAL-Pin müssen in einer Postgui-HAL-Datei als angegeben werden. Der HAL-Pin ist erst verfügbar, nachdem die QtPlasmaC-GUI geladen wurde.

Ohmscher Test

QtPlasmaC aktiviert das Ausgangssignal Ohmic Probe Enable, und wenn der Eingang der Ohmic Probe erkannt wird, leuchtet die LED-Anzeige im SENSOR-Panel auf. Der Hauptzweck dieser Funktion besteht darin, einen schnellen Test auf eine kurzgeschlossene Brennerspitze zu ermöglichen.

```
Code = ohmic-test
```

NOTE Das Aktivieren einer Benutzerschaltfläche als Ohmic Test-Schaltfläche fügt einen externer HAL-Pin hinzu, der von einem Anhänger (engl. pendant) usw. verbunden werden kann. HAL-Verbindungen zu diesem HAL-Pin müssen in einer Postgui-HAL-Datei als angegeben werden Der HAL-Pin ist erst verfügbar, nachdem die QtPlasmaC-GUI geladen wurde.

Schnittart (engl. cut type)

Mit dieser Schaltfläche können Sie zwischen den beiden cut types, Durchstechen und Schneiden (Standardmodus) oder Nur Durchstechen, umschalten.

```
Code = cut-type
```

Verbrauchsmaterialien wechseln

Durch Drücken dieser Taste wird der Brenner bei angehaltener Maschine zu den angegebenen Koordinaten bewegt, um dem Benutzer einen einfachen Zugang zum Wechseln der Brennerverbrauchsmaterialien zu ermöglichen.

Gültige Einträge sind X*nnn* Y*nnn* F*nnn*. Mindestens eine der X- oder Y-Koordinaten ist erforderlich, die Vorschubgeschwindigkeit (F) ist optional.

Die X- und Y-Koordinaten werden in absoluten Maschinenkoordinaten angegeben. Wenn X oder Y fehlen, wird die aktuelle Koordinate für diese Achse verwendet.

Die Vorschubgeschwindigkeit (engl. feed rate) (F) ist optional, wenn es fehlt oder ungültig ist, dann wird die Vorschubgeschwindigkeit des aktuellen Materials verwendet.

Es gibt drei Methoden, um in die vorherigen Koordinaten zurückzukehren:

1. Drücken Sie erneut die Taste **Verbrauchsmaterial wechseln** (engl. change consumables) - der Brenner kehrt zu den ursprünglichen Koordinaten zurück und das Gerät wartet in dieser Position, bis der Benutzer das Programm fortsetzt.
2. Drücken Sie **CYCLE RESUME** - der Brenner kehrt zu den ursprünglichen Koordinaten zurück und das Programm wird fortgesetzt.
3. Drücken Sie **CYCLE STOP** - der Brenner wird zu den ursprünglichen Koordinaten zurückzukehren und das Programm wird abgebrochen.

```
Code = change-consumables X10 Y10 F1000
```

NOTE Das Aktivieren einer Benutzerschaltfläche als Schaltfläche "Change Consumables" fügt einen externen HAL-Pin hinzu, der von einem Hängegerät usw. aus angeschlossen werden kann. HAL-Verbindungen zu diesem HAL-Pin müssen in einer Postgui-HAL-Datei angegeben werden, da der HAL-Pin erst verfügbar ist, wenn die QtPlasmaC-GUI geladen wurde.

Laden

Das Laden eines G-Code-Programms aus dem Verzeichnis, das durch die Variable **PROGRAM_PREFIX** in der Datei <*Maschinenname*>.ini angegeben ist (normalerweise ~/linuxcnc/nc_files), ist mit dem folgenden Format möglich:

```
Code = load G-code.ngc
```

Wenn sich die G-Code-Datei des Benutzers in einem Unterverzeichnis des Verzeichnisses **PROGRAM_PREFIX** befindet, wird der Name des Unterverzeichnisses an den Anfang des G-Code-Dateinamens angehängt. Beispiel für ein Unterverzeichnis namens **plasma**:

```
Code = load plasma/G-code.ngc
```

Beachten Sie, dass das erste "/" nicht notwendig ist, da es automatisch hinzugefügt wird.

Brenner-Puls

Schaltet den Brenner für eine bestimmte Zeit ein. Die Zeit muss in Sekunden mit bis zu einer Dezimalstelle angegeben werden. Die maximal zulässige Zeit beträgt 3 Sekunden; alles, was über diesem Wert wird auf 3 Sekunden begrenzt. Unten ist ein Beispiel für einen 0,5-Sekunden-Impuls zu sehen.

Wird keine Zeit angegeben, so wird standardmäßig 1 Sekunde verwendet. Impulszeiten mit mehr als einer Dezimalstelle werden auf eine Dezimalstelle gerundet.

Wenn Sie die Taste während des Countdowns erneut drücken, wird der Brenner ausgeschaltet, ebenso wie durch Drücken der *Esc*-Taste, wenn Tastenkombinationen auf der Registerkarte EINSTELLUNGEN (engl. settings) aktiviert sind.

Wird die Taste vor Ablauf des Countdowns losgelassen, schaltet sich der Brenner nach Ablauf des Countdowns aus. Wird die Taste nach Ablauf des Countdowns gedrückt gehalten, bleibt der Brenner so lange eingeschaltet, bis die Taste losgelassen wird.

```
n Code = torch-pulse 0.5
```

NOTE Die Aktivierung einer Benutzertaste als Brenner Impulse (engl. torch pulse) Taste fügt einen external HAL pin hinzu, der von einem Pendant usw. angeschlossen werden kann. HAL-Verbindungen zu diesem HAL-Pin müssen in einer Postgui-HAL-Datei angegeben werden, da der HAL-Pin nicht verfügbar ist, bis die QtPlasmaC-GUI geladen ist.

Einzelschnitt

Führen Sie einen einzelnen unidirektionalen Schnitt aus. Dabei wird die automatische Funktion Single Cut verwendet.

```
n Code = single-cut
```

Einrahmung (engl. framing)

Mit der Rahmenfunktion können Sie den Brenner innerhalb eines Rechtecks bewegen, das die Grenzen des aktuellen Auftrags umschließt.

Der HAL-Pin für die Laseraktivierung (qtplasmac.laser_on) wird während der Rahmungsbewegungen eingeschaltet, und alle X/Y-Offsets für den Laserzeiger in der Datei <*machine_name*>.prefs werden auch auf die X/Y-Bewegung angewendet. Nach Abschluss der Rahmungsbewegung bewegt sich der Brenner in die Position X0 Y0, um alle angewendeten Laser-Offsets zu löschen, und qtplasmac.laser_on wird deaktiviert.

Beim Starten eines Framing-Zyklus ist es wichtig zu beachten, dass die Z-Achse standardmäßig auf eine Höhe von [AXIS_Z]MAX_LIMIT - 5 mm (0.2") bewegt wird, bevor die X/Y-Bewegung beginnt.

Die Geschwindigkeit für die XY-Bewegungen der Rahmungsbewegung kann so festgelegt werden, dass die Rahmungsbewegung immer mit einer bestimmten Geschwindigkeit erfolgt. Dies kann durch Hinzufügen der Vorschubgeschwindigkeit (F) als letzten Teil des Tastencodes erreicht werden. Wenn die Vorschubgeschwindigkeit im Schaltflächencode weggelassen wird, so wird die Geschwindigkeit der Rahmenbewegung standardmäßig auf die Vorschubgeschwindigkeit für das aktuell ausgewählte Material eingestellt.

Die folgenden GUI-Schaltflächen und Tastenkombinationen (falls in der SETTINGS Tab aktiviert) sind während der Framing-Bewegung gültig:

1. Drücken von **CYCLE STOP** oder der ESC-Taste keyboard shortcut - Stoppt die Framing-Bewegung.

2. Durch Drücken von **CYCLE PAUSE** oder der Leertaste keyboard shortcut- wird die Framing-Bewegung angehalten.
3. Drücken Sie **CYCLE RESUME** oder die Tastenkombination STRG+r keyboard shortcut- Setzt die angehaltene Framing-Bewegung fort.
4. Ändern Sie den **FEED SLIDER** oder eine der Tastenkombinationen CTRL+0-9 keyboard shortcuts - Verlangsamt die Vorschubgeschwindigkeit.

NOTE	WENN DER VORSCHUB FÜR DIE RAHMENBEWEGUNG GEÄNDERT WIRD, MUSS DER SCHIEBEREGLER FÜR DEN VORSCHUB AUF 100 % ZURÜCKGESTELLT WERDEN, BEVOR SIE DEN ZYKLUS STARTEN UND DEN GELADENEN AUFTRAG SCHNEIDEN.

```
Code = framing
```

Der Benutzer kann die anfängliche Standard-Z-Bewegung auslassen und die Rahmungssequenz mit der aktuellen Z-Höhe ausführen, indem er "usecurrentzheight" nach "framing" hinzufügt.

```
Code = framing usecurrentzheight
```

Um eine Vorschubgeschwindigkeit festzulegen:

```
Code = framing F100
```

oder:

```
Code = framing usecurrentzheight F100
```

Die Aktivierung einer Benutzertaste als Rahmentaste fügt einen external HAL pin hinzu, der von einem Hängegerät usw. angeschlossen werden kann. HAL-Verbindungen zu diesem HAL-Pin müssen in einer Postgui-HAL-Datei angegeben werden, da der HAL-Pin erst verfügbar ist, wenn die QtPlasmaC-GUI geladen ist.

Manueller Schnitt

Manueller Schnitt funktioniert genauso wie die Taste **F9**, um einen manuellen Einzelschnitt zu beginnen oder zu beenden.

```
Code = manual-cut
```

Offset-Viewer

Dies ermöglicht das Ein- und Ausblenden eines Offset-Anzeigebildschirms, der alle Maschinenoffsets anzeigt. Alle relativen Versätze können bearbeitet werden und die Koordinaten des Arbeitssystems G54 ~ G59.3 können mit eigenen Namen versehen werden.

```
Code = offsets-view
```

Letzte Datei laden

Dies ermöglicht das Laden der zuletzt geänderten Datei in einem Verzeichnis. Die Angabe des Verzeichnisnamens ist optional, und wenn er weggelassen wird, so wird standardmäßig das letzte Verzeichnis verwendet, aus dem eine Datei geladen wurde.

```
Code = latest-file /home/me/linuxcnc/nc_files/qtplasmac-test
```

Benutzerhandbuch

Dies ermöglicht die Darstellung/Verhitzung des Online-HTML-Benutzerhandbuchs spezifisch für die Version des gerade laufenden LinuxCNC. Beachten Sie, dass ein Zugang zum Internet für diese Funktionalität erforderlich ist.

```
Code = user-manual
```

Wechsel zu/von Joint Modus

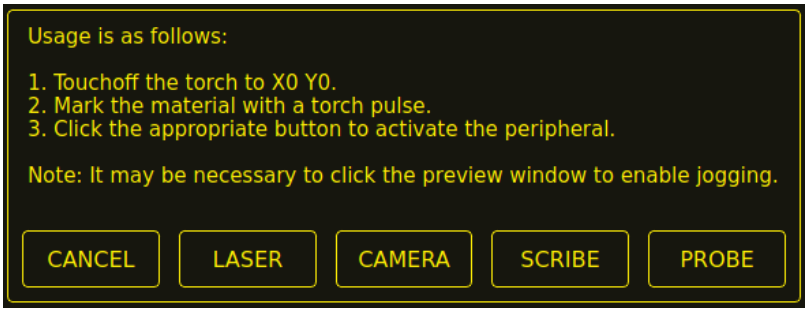
Dieser Button ermöglicht das Umschalten zwischen Joint-Modus und Teleop-Modus. Damit dieser Button aktiv ist, muss die Maschine eingeschaltet und referenziert (engl. homed) sein.

```
Code = toggle-joint
```

Periphere Offsets (Laser, Kamera, Ritzer, Offset-Tastkopf)

Verwenden Sie die folgende Sequenz, um die Offsets für einen Laser, eine Kamera, einen Ritzer oder einen Offset-Taster einzustellen:

1. Legen Sie ein Stück Restmaterial unter den Brenner.
2. Die Maschine muss referenziert und im Leerlauf sein, bevor Sie fortfahren.
3. Öffnen Sie die Registerkarte SETTINGS.
4. Klicken Sie auf die Schaltfläche SET OFFSETS, um das Dialogfeld Peripherie-Offsets einstellen zu öffnen.

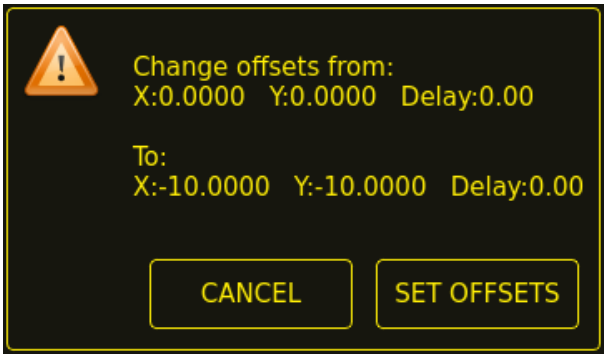


5. Klicken Sie auf die Schaltfläche X0Y0, um die Brennerposition auf Null zu setzen.
6. Markieren Sie das Material auf eine der folgenden Arten:
   a. Bewegen Sie den Brenner nach unten, um die Lochstechhöhe zu erreichen, und pulsieren Sie dann den Brenner, um eine Vertiefung im Material zu erzeugen.
   b. Geben Sie die Markierungsfarbe auf den Brennerschild und bewegen Sie den Brenner nach unten, um das Material zu markieren.

7. Klicken Sie auf die entsprechende Schaltfläche, um das Peripheriegerät zu aktivieren.
8. Das Dialogfeld Peripherie-Offsets abrufen wird nun angezeigt.



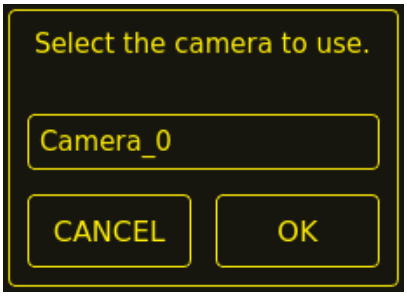
9. Heben Sie die Z-Achse an, so dass der Brenner und das Peripheriegerät vom Material entfernt sind.
10. Bewegen Sie die X/Y-Achsen so, dass das Peripheriegerät in der Markierung des Brenners zentriert ist.
11. Klicken Sie auf die Schaltfläche GET OFFSETS, um die Offsets zu erhalten, und ein Bestätigungsdialog wird geöffnet.



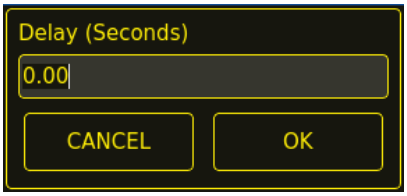
12. Klicken Sie auf SET-OFFSETS und die Offsets werden nun gespeichert.

Sie können den Vorgang jederzeit abbrechen, indem Sie auf die Schaltfläche ABBRUCH drücken, wodurch der Dialog geschlossen wird und keine Änderungen gespeichert werden.

Wenn unter Punkt 7 CAMERA ausgewählt wurde und mehr als eine Kamera vorhanden ist, wird ein Dialogfeld zur Kameraauswahl angezeigt. Die entsprechende Kamera muss ausgewählt werden, bevor der Dialog "Peripherie-Offsets abrufen" angezeigt wird.



Wenn PROBE unter Punkt 7 ausgewählt wurde, wird vor dem Bestätigungsdialog unter Punkt 11 ein Verzögerungsdialog angezeigt. Dies ist die Verzögerung, die erforderlich ist, um die Sonde in ihre Arbeitsposition zu bringen.



NOTE Es kann notwendig sein, auf das Vorschaufenster zu klicken, um das Rütteln zu aktivieren. Durch das obige Verfahren die Offsets sind für den Einsatz sofort verfügbar und es ist kein Neustart von LinuxCNC erforderlich.

Z-Bewegung beibehalten

Standardmäßig entfernt QtPlasmaC alle Z-Bewegungen aus einer geladenen G-Code-Datei und fügt eine anfängliche Z-Bewegung hinzu, um den Brenner am Anfang der Datei in die Nähe des oberen Endes des Verfahrwegs zu bringen. Wenn der Benutzer seinen Tisch mit einem im Brennerhalter montierten Marker, Schleppmesser, Diamantritzel usw. verwenden möchte, kann QtPlasmaC die Z-Bewegungen beim Ausführen eines Programms beibehalten, indem es den folgenden Befehl in eine G-Code-Datei einfügt:

```
#<keep-z-motion> = 1
```

Das kann auf zwei Arten genutzt werden: . In a G-code file with no M3 (cutting, scribing, or spotting) commands. In this case, #<keep-z-motion> = 1 can be placed anywhere before the first Z movement and all subsequent Z motion will be retained. The G-code filter will not add any initial Z movements. . In a G-code file that also contains subsequent spotting and/or cutting operations. In this case, the marking portion of the file that contains Z movements needs to preceed the spotting and/or cutting operations, and the marking section needs to have #<keep-z-motion> = 1 at the beginning and #<keep-z-motion> = 0 at the end. In this case, the G-code filter will automatically add an initial Z movement for the first M3 command after #<keep-z-motion> = 0.

In either case, M3 commands are not supported while #<keep-z-motion> is active.

Omitting #<keep-z-motion>, or setting #<keep-z-motion> to anything but 1 will cause QtPlasmaC the default behavior of stripping all Z motion from a loaded G-code file and adding an initial Z movement to bring the torch near the top of travel before the first M3 command.

Externe HAL-Pins

QtPlasmaC erstellt einige HAL-Pins, die für den Anschluss eines externen Tasters oder einer Fernbedienung usw. verwendet werden können.

HAL-Verbindungen zu diesen HAL-Pins müssen in einer Postgui-HAL-Datei spezifiziert werden, da die HAL-Pins nicht verfügbar sind, bis die QtPlasmaC-GUI geladen ist.

Die folgenden HAL-Bit-Pins werden immer erzeugt. Der HAL-Pin hat das gleiche Verhalten wie der zugehörige QtPlasmaC GUI-Button.

Benutzer Button Funktion	HAL-Pin	GUI Funktion
Maschinenleistung umschalten	qtplasmac.ext_power	POWER (engl. für Leistung oder Strom)
Ausführen des geladenen G-Code-Programms	qtplasmac.ext_run	ZYKLUSSTART
Pause/Fortsetzen des geladenen G-Code-Programms	qtplasmac.ext_pause	ZYKLUS PAUSE/ZYKLUS FORTSETZEN
Pausieren des geladenen G-Code-Programms	qtplasmac.ext_pause_only	ZYKLUSPAUSE
Fortsetzen des geladenen G-Code-Programms	qtplasmac.ext_resume	ZYKLUS FORTSETZUNG
Abbruch des geladenen G-Code-Programms	qtplasmac.ext_abort	ZYKLUS STOP (engl. cycle stop)
Touchoff X- und Y-Achsen auf Null	qtplasmac.ext_touchoff	X0Y0
Verwenden eines Lasers zum Festlegen eines Ursprungs mit oder ohne Drehung	qtplasmac.ext_laser_touchoff	LASER
Umsprung/Wechsel des qtplasmac.laser_on Pin	qtplasmac.ext_laser_toggle	k.A.
Ausführen/Anhalten/Fortsetzen des geladenen G-Code-Programms	qtplasmac.ext_run_pause	ZYKLUS-START, ZYKLUS-PAUSE, ZYKLUS-WIEDERAUFNAHME nacheinander
Höhenverstellung des Brenners plus	qtplasmac.ext_height_ovr_plus	OVERRIDE
Übersteuerung der Brennerhöhe minus	qtplasmac.ext_height_ovr_minus	OVERRIDE -
Brennerhöhen-Override zurückgesetzt	qtplasmac.ext_height_ovr_reset	OVERRIDE RESET TO 0.00
Übersteuerungsskala für die Brennerhöhe	qtplasmac.ext_height_ovr_scale	k.A.
Umschalten der Jogginggeschwindigkeit zwischen schnell und langsam	qtplasmac.ext_jog_slow	SCHNELL/LANGSAM JOGGEN
THC ein-/ausschalten	qtplasmac.ext_thc_enable	THC AKTIVIEREN
Brenner ein-/ausschalten	qtplasmac.ext_torch_enable	BRENNER AKTIVIEREN

Benutzer Button Funktion	HAL-Pin	GUI Funktion
Umschalten Ecke Sperre aktivieren	qtplasmac.ext_cornerlock_enable	VELOCITY ANTI DIVE ENABLE
Voidlock-Freigabe umschalten	qtplasmac.ext_voidlock_enable	VOID ANTI DIVE ENABLE
Wechsel auto-Volts ein/aus	qtplasmac.ext_auto_volts_enable	AUTO VOLTS
Ohmsche Sonde ein-/ausschalten	qtplasmac.ext_ohmic_probe_enable	OHMISCH AKTIVIEREN
Mesh Modus wechseln	qtplasmac.ext_mesh_mode	MESH-MODUS
Umschalten Lichtbogen ignorieren OK	qtplasmac.ext_ignore_arc_ok	IGNORE OK
Vorwärts entlang des programmierten Pfades	qtplasmac.ext_cutrec_fwd	CUT RECOVERY FWD
Rückwärts entlang des programmierten Pfads	qtplasmac.ext_cutrec_rev	CUT RECOVERY REV
Abbrechen einer Schnittwiederherstellungsbewegung	qtplasmac.ext_cutrec_cancel	CUT RECOVERY CANCEL MOVE
Nach oben bewegen	qtplasmac.ext_cutrec_n	SCHNITT FORTSETZUNG (engl. cut recovery) Pfeil nach oben
Nach unten bewegen	qtplasmac.ext_cutrec_s	CUT RECOVERY Pfeil nach unten
Nach rechts bewegen	qtplasmac.ext_cutrec_e	CUT RECOVERY Pfeil rechts
Nach links bewegen	qtplasmac.ext_cutrec_w	CUT RECOVERY Pfeil links
Nach oben-rechts bewegen	qtplasmac.ext_cutrec_ne	CUT RECOVERY Pfeil nach oben-rechts
Nach oben-links bewegen	qtplasmac.ext_cutrec_nw	CUT RECOVERY Pfeil nach oben-links
Nach rechts-unten bewegen	qtplasmac.ext_cutrec_se	CUT RECOVERY Pfeil nach unten-rechts
Nach unten nach links bewegen	qtplasmac.ext_cutrec_sw	CUT RECOVERY Pfeil nach unten-links

Die folgenden HAL-Stifte ermöglichen die Verwendung eines MPG zur Steuerung der Höhenüberwindung und werden immer erstellt.

Funktion	HAL-Pin
MPG-Höhenkontrolle einschalten	qtplasmac.ext_height_ovr_count_enable
MPG Höhe ändern	qtplasmac.ext_height_ovr_counts

Die folgenden HAL-Bitpins werden nur erstellt, wenn die Funktion in einem Benutzer-definierter Button angegeben ist. Der HAL-Pin hat das gleiche Verhalten wie die zugehörige benutzerdefinierte Button.

Benutzer Button Funktion	HAL-Pin
Sonden-Test	qtplasmac.ext_probe
Brenner-Puls	qtplasmac.ext_pulse
Ohmscher Test	qtplasmac.ext_ohmic
Verbrauchsmaterialien wechseln	qtplasmac.ext_consumables
Einrahmung (engl. framing)	qtplasmac.ext_frame_job

Die folgenden HAL-Bit-Ausgangspins werden immer erstellt und können entweder von den benutzerdefinierten Tasten Toggle HAL Pin oder Pulse HAL Pin verwendet werden, um den Zustand eines Ausgangs zu ändern.

HAL-Pin
qtplasmac.ext_out_0
qtplasmac.ext_out_1
qtplasmac.ext_out_2

Programm-Buttons ausblenden

Wenn der Benutzer über externe Tasten und/oder einen Pendant verfügt, der eine der Programmtasten CYCLE START, CYCLE PAUSE oder CYCLE STOP emuliert, ist es möglich, einige oder alle dieser GUI-Programmtasten auszublenden, indem man den folgende Optionen in den Abschnitt **[GUI_OPTIONS]** der Datei <*machine_name*>.prefs einfügt:

```
Hide run = True
Hide pause = True
Hide abort = True
```

Bei den 16:9- oder 4:3-GUIs werden durch das Ausblenden jeder dieser GUI-Schaltflächen zwei weitere benutzerdefinierte Schaltflächen in der GUI sichtbar.

Tuning-Modus 0 Arc OK

Modus 0 Arc OK basiert auf der Lichtbogenspannung, um das Arc OK-Signal zu setzen. Dies wird durch Abtasten der Lichtbogenspannung in jedem Servogewindezyklus erreicht. Damit das Lichtbogen-OK-

Signal gesetzt wird, muss eine bestimmte Anzahl aufeinander folgender Abtastungen vorliegen, die alle innerhalb eines bestimmten Schwellenwerts liegen. Diese Spannungen müssen auch innerhalb eines bestimmten Bereichs liegen.

Es gibt zwei Einstellungen in der Registerkarte PARAMETER für die Einstellung des Bereichs, diese sind:

- **OK High Volts**, das ist der obere Wert des Spannungsbereichs. Der Standardwert ist 250 V.
- **OK Low Volts**, das ist der untere Wert des Spannungsbereichs. Der Standardwert ist 60 V.

Beide Werte können durch direkte Eingabe oder mit Hilfe der Tasten zum Erhöhen/Verringern geändert werden.

Es gibt auch zwei HAL-Pins, die dem Benutzer ermöglichen, den Sollwert abzustimmen. Diese HAL-Pins sind:

- `plasmac.arc-ok-counts`, d. h. die Anzahl der aufeinanderfolgenden Messwerte innerhalb des Schwellenwerts, die erforderlich sind, um das Signal Arc OK zu setzen. Der Standardwert ist 10.
- `plasmac.arc-ok-threshold`, das ist die maximale Spannungsabweichung, die für eine gültige Spannung zulässig ist, um das Lichtbogen-OK-Signal zu setzen. Der Standardwert ist 10.

Im folgenden Beispiel wird die Anzahl der erforderlichen gültigen aufeinanderfolgenden Messwerte auf 6 festgelegt:

```
setp plasmac.arc-ok-counts 6
```

Wenn diese Einstellungen verwendet werden, sollten sie in der Datei custom.hal der Konfiguration enthalten sein.

Verlorener Lichtbogen-Verzögerung

Bei einigen Plasmastromquellen/Maschinenkonfigurationen kann es vorkommen, dass das Lichtbogen-OK-Signal entweder kurzzeitig während eines Schnittes oder dauerhaft gegen Ende eines Schnittes verloren geht, was dazu führt, dass QtPlasmaC das Programm anhält und einen Fehler "Gültiger Lichtbogen verloren" meldet.

Es gibt einen HAL-Pin mit dem Namen `plasmac.arc-lost-delay`, mit dem eine Verzögerung (in Sekunden) eingestellt werden kann, die ein angehaltenes Programm/einen Fehler verhindert, wenn das verlorene Arc-OK-Signal wiedergewonnen oder der **M5**-Befehl erreicht wird, bevor die eingestellte Verzögerungszeit abgelaufen ist.

Es ist wichtig zu beachten, dass die THC deaktiviert und in der Schnitthöhe verriegelt ist, in der das Lichtbogen-OK-Signal verloren ging.

Der folgende Code würde eine Verzögerung von 0,1 Sekunden einstellen:

```
setp plasmac.arc-lost-delay 0.1
```

Es wird empfohlen, dass der Benutzer diese PIN in der Datei custom.hal festlegt.

Diese Einstellung sollte nur verwendet werden, wenn der Benutzer die oben genannten Symptome feststellt. Es sollte auch beachtet werden, dass der Benutzer die entsprechenden Ignore Arc OK G-Code-Befehle verwenden könnte, um ein ähnliches Ergebnis zu erzielen.

Null-Fenster

Geringe Schwankungen der angezeigten Lichtbogenspannung im Leerlauf der Maschine sind möglich und hängen von vielen verschiedenen Variablen ab (elektrisches Rauschen, falsche THCAD-Einstellung usw.).

Wenn nach der Beseitigung aller Einflussfaktoren immer noch eine kleine Schwankung vorhanden ist, kann diese durch Vergrößerung des Spannungsfensters, in dem QtPlasmaC 0 V anzeigt, beseitigt werden.

Der Pin zur Einstellung dieses Wertes heißt plasmac.zero-window und ist standardmäßig auf 0,1 eingestellt. Um diesen Wert zu ändern, fügen Sie den Pin und den gewünschten Wert in die Datei custom.hal ein.

Das folgende Beispiel würde das Spannungsfenster so einstellen, dass es von -5 V bis +5 V als 0 V angezeigt wird:

```
setp plasmac.zero-window 5
```

Void-Erkennung optimieren

Zusätzlich zur Einstellung **Void Slope** in der Reigsterkarte PARAMETER gibt es zwei HAL-Pins, die bei der Feinabstimmung des Void-Anti-Dive helfen. Diese HAL-Pins sind:

- **plasmac.void-on-cycles**, d.h. die Anzahl der Überschreitungen der Steigungsrate, die erforderlich sind, um die Anti-Void-Funktion zu aktivieren. Der Standardwert ist 2.
- **plasmac.void-off-cycles**, d. h. die Anzahl der Zyklen, in denen die Steigungsrate nicht überschritten wird, um das Anti-Void-Verfahren zu deaktivieren. Der Standardwert ist 10.

Im folgenden Beispiel wird die Anzahl der erforderlichen Einschaltzyklen auf 3 festgelegt:

```
setp plasmac.void-on-cycles 3
```

Ziel ist es, einen möglichst niedrigen Wert für die Leerlaufsteigung zu haben, ohne dass es zu Fehlauslösungen kommt, und dann die Ein- und Ausschaltzyklen so anzupassen, dass eine saubere Aktivierung und Deaktivierung der Leerlaufsperre gewährleistet ist. In den meisten Fällen sollte es nicht notwendig sein, die Ein- und Ausschaltzyklen gegenüber dem Standardwert zu ändern.

Wenn diese Einstellungen verwendet werden, sollten sie in der Datei custom.hal der Konfiguration enthalten sein.

Max. Versatz (engl. offset)

Max Offset ist der Abstand (in Millimetern) von der Z MAX_LIMIT, den QtPlasmaC der Z-Achse erlaubt,

während sie unter Maschinensteuerung steht.

Der Pin für die Einstellung dieses Wertes heißt plasmac.max-offset und der Standardwert (in Millimetern) ist auf 5 eingestellt. Um diesen Wert zu ändern, fügen Sie den Pin und den gewünschten Wert in die Datei custom.hal ein. Es wird nicht empfohlen, Werte unter 5 mm zu verwenden, da eine Überschreitung des Offsets zu unvorhergesehenen Problemen führen kann.

Das folgende Beispiel würde den Abstand von Z MAX_LIMIT auf 10 mm setzen:

```
setp plasmac.max-offset 10
```

Aktivieren von Registerkarten bei automatischer Bewegung

Standardmäßig sind alle Registerkarten außer der MAIN Tab während der automatisierten Bewegung deaktiviert. Es ist möglich, alle Registerkarten außer der CONVERSATIONAL Tab während der automatisierten Bewegung zu aktivieren, indem man den folgenden HAL-Pin True setzt:

```
setp qtplasmac.tabs_always_enabled 1
```

WARNING Es liegt in der Verantwortung des Bedieners sicherzustellen, dass die Maschine mit einem geeigneten, funktionierenden Hardware-Notaus (engl. E-stop) ausgestattet ist. Wenn nur ein Touchscreen zur Navigation in der QtPlasmaC-GUI verwendet wird, gibt es keine Möglichkeit, die automatische Maschinenbewegung auf einer anderen Registerkarte als der MAIN-Registerkarte anzuhalten.

Aufhebung der Jog-Sperre über Z+ Jog

Es ist möglich, die Jog-Sperre außer Kraft zu setzen, indem man die GUI oder die Tastatur benutzt, um in die Z+ Richtung zu joggen, anstatt das Feld Override Jog auf der SETTINGS Registerkarte zu markieren.

Dazu müssen Sie die folgende Option in der Datei <*Maschinenname*>.prefs im Ordner <Maschinenname> unter **[GUI_OPTIONS]** auf **True** setzen:
`Override jog inhibit via Z+`

QtPlasmaC Status-Ausgänge

Die HAL-Komponente plasmac verfügt über einen HAL-Pin namens **plasmac.state-out**, der als Schnittstelle zu benutzercodierten Komponenten verwendet werden kann, um den aktuellen Zustand der Komponente zu ermitteln.

Table 79. Verschiedene Zustände, die QtPlasmaC annehmen kann

Zustand	Name	Beschreibung
0	IDLE	im Leerlauf und wartet auf einen Startbefehl
1	PROBE_HEIGHT	nach unten auf Sondenhöhe fahren

Zustand	Name	Beschreibung
2	PROBE_DOWN	Sonde absenken, bis Material erkannt wird
3	PROBE_UP	Sonde nach oben, bis das Material nicht mehr erfasst wird, dadurch wird die Nullhöhe eingestellt
4	ZERO_HEIGHT	zur Zeit nicht genutzt
5	PIERCE_HEIGHT	nach oben auf die Pierce-Höhe fahren
6	TORCH_ON	Brenner anschalten
7	ARC_OK	warten, bis Lichtbogen OK erkannt wird
8	PIERCE_DELAY	Wartezeit für Durchstichverzögerung
9	PUDDLE_JUMP	xy-Bewegung beginnt, auf Pfützen-Sprunghöhe gehen
10	CUT_HEIGHT	auf Schnitthöhe fahren
11	CUT_MODE_01	Schneiden in Modus 0 oder Modus 1
12	CUT_MODE_2	Schneiden im Modus 2
13	PAUSE_AT_END	Bewegungspause am Ende des Schnitts
14	SAFE_HEIGHT	auf sichere Höhe bringen
15	MAX_HEIGHT	auf maximale Höhe fahren
16	END_CUT	den aktuellen Schnitt beenden
17	END_JOB	den laufenden Auftrag beenden
18	TORCHPULSE	ein brennerimpuls ist aktiv
19	PAUSED_MOTION	Die Wiederherstellungsbewegung des Schnitts ist während der Pause aktiv.
20	OHMIC_TEST	ein ohmscher Test ist aktiv
21	PROBE_TEST	Ein Sondentest ist aktiv
22	SCRIBING	ein Scribing-Job ist aktiv
23	CONSUMABLE_CHANGE_ON	zu den Koordinaten für den Wechsel des Verbrauchsmaterials gehen
24	CONSUMABLE_CHANGE_OFF	Rückkehr von Verbrauchsmaterial-Änderungskoordinaten
25	CUT_RECOVERY_ON	Schnittfortsetzung is aktiv
26	CUT_RECOVERY_OFF	Schnittfortsetzung ist deaktiviert
27	DEBUG	debug state, for testing purposes only

Der DEBUG-Zustand dient nur zu Testzwecken und wird normalerweise nicht angetroffen.

QtPlasmaC Debug Print

Die HAL-Komponente plasmac hat einen HAL-Pin mit dem Namen **plasmac.debug-print**, der, wenn er auf 1 (true) gesetzt ist, jede Zustandsänderung als Debug-Hilfe auf dem Terminal ausgibt.

Hypertherm PowerMax Kommunikation

Die Kommunikation kann mit einem Hypertherm PowerMax-Plasmaschneider mit RS485-Anschluss hergestellt werden. Diese Funktion ermöglicht die automatische Einstellung von **Schneidmodus**, **Schneidstromstärke** und **Gasdruck** anhand der **Schneidparameter** der Materialdatei. Darüber hinaus kann der Benutzer die **Arc On Time** des PowerMax im Format hh:mm:ss auf der Registerkarte STATISTIKEN anzeigen.

Wenn **Gasdruck** auf Null eingestellt ist, berechnet der PowerMax automatisch den erforderlichen Druck aus **Schneidmodus**, **Schneidstrom**, Brennertyp und Brennerlänge.

Wenn Sie den Schneidmodus ändern, wird der Gasdruck auf Null gesetzt, so dass die Maschine ihren automatischen Gasdruckmodus verwendet.

Die Höchst- und Mindestwerte dieser Parameter werden vom Plasmaschneider abgelesen und die entsprechenden Drehknöpfe in den Schnittparametern werden dann durch diese Werte begrenzt. Der Gasdruck kann nicht von Null aus geändert werden, bis die Kommunikation hergestellt ist.

Diese Funktion wird aktiviert, indem der richtige Anschlussname für die Option PM_PORT im Abschnitt **[POWERMAX]** der Datei <*Maschinenname*>.prefs gesetzt wird. Wenn die Option PM_PORT in der Datei <*Maschinenname*>.prefs nicht gesetzt ist, werden die mit dieser Funktion verbundenen Widgets nicht sichtbar sein.

Beispiel für die Aktivierung der Hypertherm PowerMax-Kommunikation auf USB0:

```
[POWERMAX]
Port = /dev/ttyusb0
```

Wenn der Benutzer den Namen des Ports nicht kennt, gibt es ein Python-Skript im Konfigurationsverzeichnis, das alle verfügbaren Ports anzeigt und auch verwendet werden kann, um die Kommunikation mit der Plasmaeinheit zu testen, bevor diese Funktion in der QtPlasmaC-GUI aktiviert wird.

Um das Testskript zu verwenden, folgen Sie diesen Anweisungen:

Geben Sie für eine Paketinstallation (Buildbot) den folgenden Befehl in einem Terminalfenster ein:

```
pmx485-test
```

Geben Sie für eine "run in place"-Installation die folgenden beiden Befehle in ein Terminalfenster ein:

```
source ~/linuxcnc-dev/scripts/rip-environment
```

```
pmx485-test
```

Die Anzeige der Gasdruckeinheiten (psi oder bar) wird durch die bei der Ersteinrichtung der Kommunikationsverbindung empfangenen Daten bestimmt und wird dann neben der Einstellung für den Gasdruck im Abschnitt MATERIAL auf der PARAMETERS Registerkarte angezeigt.

Die PowerMax-Maschine wechselt nach dem Aufbau der Kommunikation in den Remote-Modus und kann zu diesem Zeitpunkt nur ferngesteuert werden (über die QtPlasmaC-GUI). Die Verbindung kann durch Beobachtung des PowerMax-Displays validiert werden.

Um den PowerMax wieder in den lokalen Modus zu schalten, kann der Benutzer entweder:

1. PowerMax Comms auf der Haup-Registerkarte deaktivieren
2. LinuxCNC Schließen, wodurch der PowerMax während des Herunterfahrens in den lokalen Modus versetzt wird.
3. Den PowerMax für 30 Sekunden aus- und dann wieder einschalten.

TIP Wenn die PowerMax-Kommunikation aktiv ist, wird durch Auswahl von Mesh Mode automatisch der CPA-Modus auf dem PowerMax-Gerät ausgewählt.

NOTE Um die PowerMax-Kommunikationsfunktion zu nutzen, muss das Python-Modul pyserial installiert sein.
Wenn pyserial nicht installiert ist, wird eine Fehlermeldung angezeigt.

Um pyserial zu installieren, geben Sie den folgenden Befehl in ein Terminalfenster ein:

```
sudo apt install python3-serial
```

Ein typischer Anschlussplan ist im Anhang dieses Dokuments sowie bestätigte Arbeitsschnittstellen dargestellt.

Bewegtes Einstechen (engl. Moving Pierce)

Ein **moving pierce** ermöglicht es dem Brenner, sich während der **Pierce Delay**-Phase zu bewegen. Der Vorteil besteht darin, dass sich damit dickere Materialien durchstechen lassen, als es mit einem stationären **Pierce** möglich wäre. Außerdem kann diese Bewegung dazu beitragen, die Lebensdauer der Verschleißteile zu verlängern, da sie verhindert, dass geschmolzenes Material in die Brennerdüse zurückgespritzt wird.

Durch die Verwendung von M159 kann ein „moving pierce“ konfiguriert werden.

Die Syntax für den M159 Befehl ist wie folgt:

```
M159 Pn Qn
```

Aktionscode (engl. action code) (P)	Action	Beschreibung	Wert (Q)
601	Durchstech-Typ (engl. pierce type)	0=Normal, 1=Wackelnd (engl. wiggle), 2=Aufsteigend (engl. ramp)	0,1,2
602	Durchstich bewegungsverzögerung (engl. pierce motion delay)	Verzögerung, bevor die Z-Bewegung zur Pierce-Endhöhe beginnt. Ausgedrückt als Prozentsatz der Pierce-Verzögerung.	Ganzzahl 0 bis 100
603	Endhöhe des Durchstichs (engl. pierce end height)	Ziel-Pierce-Höhe am Ende der Pierce-Verzögerung. Sie liegt normalerweise unter der Pierce-Höhe. Angegeben in Maschineneinheiten.	Gleitkommazahl (engl. float)
604	Schnitthöhenverzögerung (engl. cut height delay)	Verzögerung am Ende des Übergangs zur Pierce-Endhöhe, bevor der Übergang zur Schnitthöhe erfolgt. Angegeben in Sekunden.	Gleitkommazahl (engl. float)
605	Ausnehm-Geschwindigkeit	Geschwindigkeit der Ausnehmung. Ausgedrückt in Maschineneinheiten/min.	Gleitkommazahl (engl. float)
606	Ausnehmabstand	Länge der Ausnehmung. Ausgedrückt in Maschineneinheiten.	Gleitkommazahl (engl. float)
607	Geschwindigkeit des Kriechgangs (engl. creep speed)	Kriechgeschwindigkeit, die nach Abschluss des Ausräumens (engl. gouge) wirksam wird. Angegeben in Maschineneinheiten.	Gleitkommazahl (engl. float)
608	Kriechweg (engl. creep distance)	Länge des Kriechwegs. Ausgedrückt in Maschineneinheiten.	Gleitkommazahl (engl. float)

Aktionscode (engl. action code) (P)	Action	Beschreibung	Wert (Q)
609	Reset	Setzt die Werte für die Aktionscodes 601–608 wieder auf 0 zurück und stellt damit das Standardverhalten her.	Nicht Erforderlich

Die folgenden moving pierce Modelle sind verfügbar:

Pendelndes Einstechen (Wiggle Pierce)

Das unterstützte Modell entspricht dem von Sheetcam erzeugten Wiggle Pierce. Bei einer geraden Anfahrbewegung (Lead-in) zum Hauptschnitt wird erwartet, dass sich der Wiggle Pierce für eine gewisse Strecke entlang der Anfahrbewegung hin- und herbewegt. Genau genommen ist diese gerade Bewegung beliebig. Technisch ist während der Piercing-Verzögerung jede X/Y-Bewegung möglich, und es liegt am CAM-Werkzeug oder am Benutzer, dies zu programmieren.

Die Einschränkung besteht darin, dass diese Bewegung während des Pierce Delay (Verzögerung) abgeschlossen sein sollte. Falls nicht, wechselt der Brenner nach Ablauf des Pierce Delay auf die normale Schnitthöhe – möglicherweise noch bevor die Wiggle-Bewegung abgeschlossen ist.

Daher müssen die Länge des Wiggle und die Vorschubgeschwindigkeit bei der Berechnung des Pierce Delay berücksichtigt werden, oder die Größe des Wiggle wird basierend auf Vorschubgeschwindigkeit und Pierce Delay eingeschränkt.

Zum Beispiel:

- Vorschubgeschwindigkeit von 1080 mm/min (18 mm/s).
- Ein Wiggle von 4 mm für 3 Pendelbewegungen.

Das bedeutet, dass die Länge des Wiggle 4 × 3 = 12 mm beträgt. Bei einer Vorschubgeschwindigkeit von 18 mm/s muss das Pierce Delay etwa 0,7 Sekunden betragen, um die Wiggle-Strecke auf Piercing-Höhe zu ermöglichen.

Der G-Code für dieses Verhalten ist:

```
M159 P601 Q1
```

Der für ein Zurücksetzen auf das Standard-Verhalten benötigte G-Code ist:

```
M159 P609
```

Ramp-Pierce

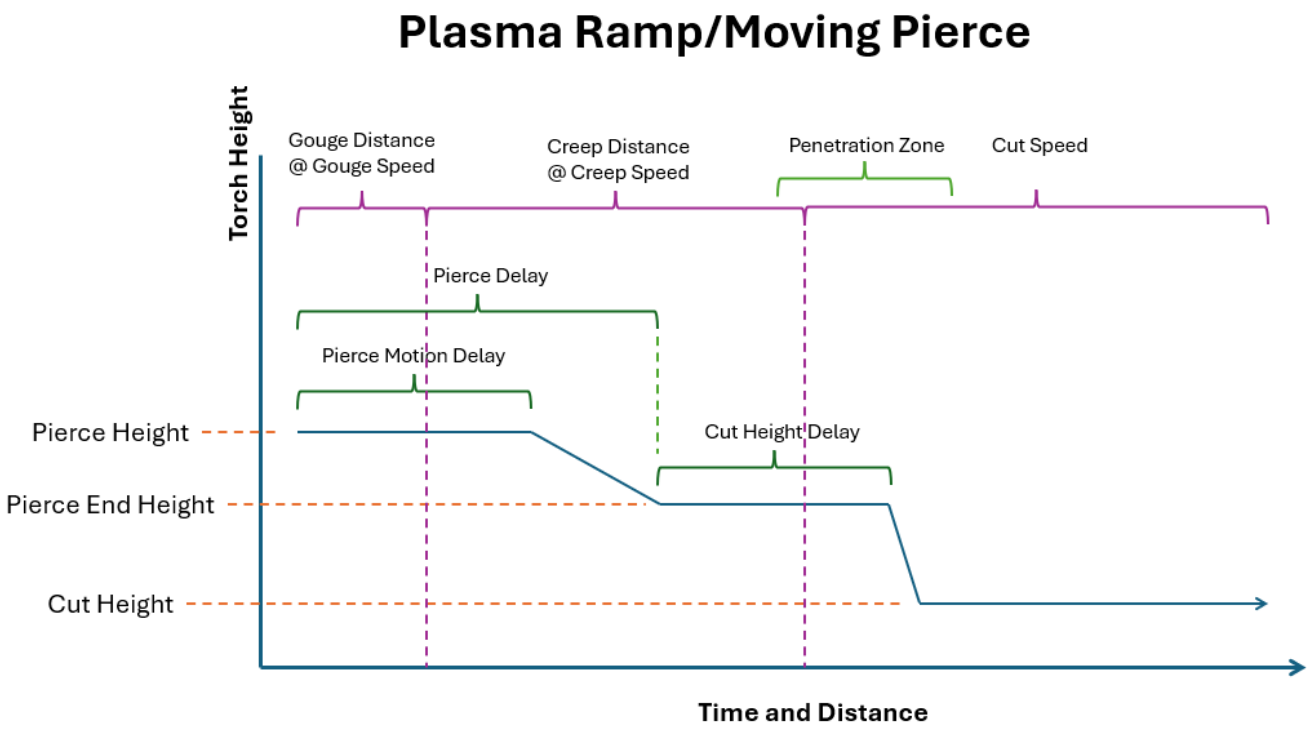
Ein ramping pierce kombiniert eine Reihe von Parametern, um eine geneigte Vertiefung zu erzeugen, die das geschmolzene Material ableitet. Die resultierende Materialausstoßung wird manchmal mit

einem „Hahnenschwanz“ verglichen, da sie sehr gerichtet erfolgt. Eine sorgfältige Gestaltung des Lead-ins ermöglicht es, das Material in eine sichere Richtung für Bediener und Maschinenkomponenten abzuleiten.

Da die Elemente zusammenwirken, um die Form des ramping pierce zu bestimmen, ist es entscheidend, alle diese Elemente sorgfältig zu berücksichtigen, wenn der ramp pierce und die ihn bestimmenden Parameter ausgelegt werden. Unvermeidlich wird es notwendig sein, Experimente durchzuführen, um Rezepte zu entwickeln, die mit der verwendeten Plasmastromquelle und dem zu schneidenden Material funktionieren.

Zu beachtende Punkte:

1. Die Ausräum (gouge)- und Kriech (creep)-Geschwindigkeiten und -strecken in Relation zur Summe des Pierce Delay des Materials und der Aktion Cut Height Delay.
2. Pierce-Höhe über dem Material sowie die Aktion der End-Pierce-Höhe im Verhältnis zur Pierce-Verzögerung, einschließlich der dabei wirksamen Geschwindigkeiten über die jeweiligen Distanzen während der Pierce-Delay-Zeit.
3. Die Schnittdiagramme des Plasmastromquellen-Herstellers für den Materialtyp und die Materialstärke.



Wie beim wiggle pierce liegt es am CAM-Tooling oder am Anwender, einen ramping pierce zu programmieren.

Es folge ein Beispiel für Code zum Einrichten eines ramp pierce:

```
(o=0,kw=2, ph=4, pd=1, ch=1.5, fr=490, th=1, cv=99, pe=0.3, jh=0, jd=0)

M159 P601 Q2
M159 P602 Q50
```

```
M159 P603 Q2.5
M159 P604 Q1
M159 P605 Q980
M159 P606 Q5
M159 P607 Q245
M159 P608 Q3
```

Dieser Code zeigt uns die folgenden Information:

1. Material magischer Kommentar.
   - Einstichhöhe (engl. pierce height) ist bei 4 mm.
   - Durchstech-Verzögerung ist 1 Sekunde.
   - Schnitthöhe (engl. cut height) ist bei 1.5 mm.
   - Schnitt-Vorschubgeschwindigkeit (engl. cut feed rate) ist bei 490mm/min.
2. Modus wird auf 2 gesetzt, dem ramp pierce.
3. Pierce Motion Delay ist 50% des Pierces Delay (0.5 Sekunden).
4. Einstich Endhöhe (engl. pierce end height) liegt bei 2.5 mm
5. Schnitthöhen Verzögerung (engl. cut height delay) liegt bei 1 Sekunde.
6. Vorschubgeschwindigkeit (engl. Original nutzte den Begriff "Gouge Speed" statt feed rate) ist bei 980 mm/min.
7. Der Verfahrweg (engl. Original nutzt den Begriff "Gauge Distance") beträgt 5 mm
8. Schleichgeschwindigkeit beträgt 245 mm/min.
9. Kriechweg (engl. creep distance) ist 3 mm.

Mit diesen Informationen kann das folgende Verhalten beschrieben werden:

Es ist wichtig zu beachten dass Beschleunigungen und Verzögerungen ausgelassen wurden bei den folgenden Berechnungen.

Der Brenner startet auf Pierce Height (4 mm über dem Material) und beginnt sich mit einer Ausräum-Geschwindigkeit von 980 mm/min (16,3 mm/s) über eine Ausräum-Strecke von 5 mm zu bewegen, was 0,3 Sekunden (5 mm / 16,3 mm/s = 0,3 s) der 0,5 s Pierce Motion Delay beansprucht.

Wenn die Ausräum-Distanz (gouge) erreicht ist, wird die Brennergeschwindigkeit auf die Kriech-Distanz (creep) von 245 mm/min (4 mm/s) für eine Kriech-Strecke von 3 mm gesetzt. Die Kriech-Strecke (creep distance) dauert etwa 0,7 Sekunden (3 mm / 4 mm/s = 0,7 s).

Der Brenner bleibt für weitere 0,2 Sekunden auf 4 mm Höhe (0,5 s Pierce Motion Delay − 0,3 s Ausräum-Distanz (gouge) bei Ausräumgeschwindigkeit = 0,2 s), danach beginnt der Brenner auf eine Pierce-Endhöhe von 2,5 mm abzusenken, über die verbleibenden 0,5 Sekunden der Pierce Delay des Materials. Da noch 0,5 Sekunden der Pierce Delay des Materials verbleiben und gleichzeitig 0,5 Sekunden bei Kriechgeschwindigkeit (creep speed) abgedeckt werden, wird die Kriech-Distanz (creep distance) gleichzeitig erreicht, wenn die Pierce-Endhöhe erreicht ist.

Wenn die Kriech-Distanz (creep distance) erreicht ist, wird die Brennergeschwindigkeit auf die

Schnittvorschubgeschwindigkeit (Cut Feed Rate) des Materials von 490 mm/min gesetzt. Da eine 1 Sekunde lange Cut Height Delay zu Beginn des Endes der Pierce Delay des Materials gestartet wurde, erfolgt der Übergang auf eine Schnitt-Höhe (Cut Height) von 1,5 mm nach Ablauf der verbleibenden 1 Sekunde Cut Height Delay.

Der obige Text soll verdeutlichen, dass eine erhebliche Menge an Konfiguration erforderlich ist und dass durch Experimente sowie die sorgfältige Verwendung unterschiedlicher Parameterkombinationen subtile Anpassungen erreicht werden können.

10.8.16. Internationalisierung

Es ist möglich, Übersetzungsdateien für QtPlasmaC zu erstellen, um sie in der Sprache des aktuellen Gebietsschemas anzuzeigen.

Um eine Übersetzungsdatei zu erstellen und oder zu bearbeiten, muss LinuxCNC installiert sein und vor Ort ausgeführt werden.

Im Folgenden wird davon ausgegangen, dass das LinuxCNC git-Verzeichnis ~/linuxcnc-dev ist.

Alle Sprachdateien werden in ~/linuxcnc-dev/share/screens/qtplasmac/languages gespeichert.

Die Datei qtplasmac.py ist eine Python-Version der GUI-Datei, die für Übersetzungszwecke verwendet wird.

Die .ts-Dateien sind die Quelldateien für die Übersetzungen. Dies sind die Dateien, die für jede Sprache erstellt/bearbeitet werden müssen.

Die .qm-Dateien sind die kompilierten Übersetzungsdateien, die von PyQt verwendet werden.

Die Verzeichnisse qtplasmac_4x3/languages und qtplasmac_9x16/languages dienen nur als Links zu den .qm-Dateien in qtplasmac/languages.

Die Sprache wird durch einen Unterstrich plus die ersten beiden Buchstaben des Gebietsschemas bestimmt, z. B. bei einer italienischen Übersetzung wäre dies `_it`. In diesem Dokument wird sie mit `_xx` bezeichnet, so dass `qtplasmac_xx.ts` in diesem Dokument für eine italienische Übersetzung eigentlich `qtplasmac_it.ts` wäre.

Das Standardgebietsschema für QtPlasmaC ist `_en`, was bedeutet, dass Übersetzungsdateien, die als `qtplasmac_en.*` erstellt wurden, nicht für Übersetzungen verwendet werden.

Wenn eines der erforderlichen Dienstprogramme (pyuic5, pylupdate5, linguist) nicht installiert ist, muss der Benutzer die benötigten Software-Tools zur Entwicklung installieren:

```
sudo apt install qttools5-dev-tools pyqt5-dev-tools
```

Wechseln Sie in das Sprachenverzeichnis:

```
cd ~/linuxcnc-dev/share/qtvcp/screens/qtplasmac/languages
```

Wenn Textänderungen an der grafischen Benutzeroberfläche vorgenommen wurden, führen Sie den folgenden Befehl aus, um die GUI-Python-Datei zu aktualisieren:

```
pyuic5 ../qtplasmac.ui > qtplasmac.py
```

Der Benutzer kann entweder eine neue Übersetzungsquelldatei für eine nicht existierende Sprachübersetzung erstellen oder eine existierende Übersetzungsquelldatei modifizieren, wenn ein Text in einer QtPlasmaC-Quelldatei geändert wurde. Wenn Sie eine bestehende Übersetzung modifizieren, die keine Änderungen in der Quelldatei erfahren hat, ist dieser Schritt nicht erforderlich.

Erstellen oder bearbeiten Sie eine .ts-Datei:

```
./langfile xx
```

NOTE Dieser Befehl ist ein Skript, das Folgendes ausführt: $ pylupdate5 **.py ..**/.py ../../../../../lib/python/qtvcp/lib/qtplasmac/*.py -ts qtplasmac_xx.ts

Die Bearbeitung der Übersetzung erfolgt mit der Anwendung Linguist:

```
linguist
```

1. Öffnen Sie die TS-Datei und übersetzen Sie die Zeichenfolgen

Es ist nicht notwendig, für jede Textzeichenfolge eine Übersetzung bereitzustellen. Wenn für eine Zeichenfolge keine Übersetzung angegeben ist, wird die ursprüngliche Zeichenfolge in der Anwendung verwendet. Der Benutzer muss auf die Länge der Zeichenketten achten, die in den Widgets erscheinen, da der Platz begrenzt ist. Wenn möglich, sollte die Übersetzung nicht länger als das Original sein.

Wenn die Bearbeitung abgeschlossen ist, speichern Sie die Datei:
`Datei -> Speichern` (engl. File → Save)

Erstellen Sie dann die .qm-Datei:
`Datei -> Freigabe` (engl. `File -> Release`)

Linguist schließen.

Dann erstellen Sie Links zu der kompilierten .qm Datei für die anderen QtPlasmaC GUIs.

```
$ ./langlink xx
```

NOTE Dieser Befehl ist ein Skript, das in sowohl qtplasmac_4x3/languages als auch in qtplasmac_9x16/languages einen Link auf die oben genannte .qm-Datei erstellt und den Link anschließend so umbenennt, dass er dem GUI-Namen entspricht.

QtPlasmaC wird beim nächsten Start in die Sprache des aktuellen Gebietsschemas übersetzt, solange eine .qm Datei in dieser Sprache existiert.

Benutzer sind eingeladen, Übersetzungsdateien für die Aufnahme in QtPlasmaC einzureichen. Eine

einfache Methode ist, die aktuelle qtplasmac_xx.ts-Datei im Forum zu posten; die Maintainer werden die Übersetzungen dann installieren.

Die bevorzugte Methode ist das Erstellen eines Pull Requests vom eigenen GitHub-Konto, wie beschrieben in der Dukumentation zum Beitragen zu LinuxCNC. Die einzigen Dateien, die eingereicht werden müssen, sind qtplasmac_xx.ts und qtplasmac_xx.qm im qtplasmc/languages Verzeichnis plus den Links in sowohl dem qtplasmac_4x3/languages und dem qtplasmac_9x16/languages Verzeichnis.

10.8.17. Anhang

Beispielkonfigurationen

Es gibt Beispielkonfigurationsdateien, um mit der QtPlasmaC-GUI Plasmaschneidmaschinen zu simulieren.

Sie können in der LinuxCNC-Auswahl unter gefunden werden: Beispielkonfigurationen → sim → qtplasmac

Drei Versionen sind sowohl in metrischen als auch in imperialen Einheiten erhältlich:

1. qtplasmac_l - 16:9-Format, Mindestauflösung 1366x768
2. qtplasmac_p - 9:16-Format, Mindestauflösung 786x1366
3. qtplasmac_s - 4:3-Format, Mindestauflösung 1024x768

Jede Beispielkonfiguration enthält ein Popup-Bedienfeld, mit dem verschiedene Eingaben in die grafische Benutzeroberfläche simuliert werden können, z. B:

1. LICHTBOGENSPANNUNG (engl. arc voltage)
2. OHMIC SENSE
3. SCHWIMMERSCHALTER
4. ABREISSSCHALTER
5. ESTOP (engl. für Notaus)

NGC Beispiele

Im Verzeichnis ~/linuxcnc/nc_files/examples/plasmac befinden sich einige Beispiel-G-Code-Dateien.

QtPlasmaC-spezifische G-Codes

Beschreibung	Code
Begin cut	`M3 $0 S1`
End cut	`M5 $0`
Begin scribe	`M3 $1 S1`

Beschreibung	Code
End scribe	`M5 $1`
Begin center spot	`M3 $2 S1`
End center spot	`M5 $2`
Alle oben genannten Punkte beenden.	`M5 $-1`
Festlegung des Materials.	`` `M190 P`*n* `` *n* steht für die Werkstoffnummer.
Warten auf Material Änderungsbestätigung.	`` `M66 PG L3 Q`*n* `` *n* ist die Verzögerungszeit (in Sekunden). Bei sehr großen Materialdateien muss dieser Wert möglicherweise erhöht werden.
Vorschubgeschwindigkeit von material ableiten.	`F#<_hal[plasmac.cut-feed-rate]>`
Enable Ignore Arc OK	`M62 P1` (synchronisiert mit der Bewegung) `M64 P1` (sofort)
Disable (deaktivieren) Ignore Arc OK	`M63 P1` (synchronisiert mit der Bewegung) `M65 P1` (sofort)
Disable (deaktivieren) THC	`M62 P2` (synchronisiert mit der Bewegung) `M64 P2` (sofort)
Enable (aktivieren) THC	`M63 P2` (synchronisiert mit der Bewegung) `M65 P2` (sofort)
Disable (deaktivieren) Torch	`M62 P3` (synchronisiert mit der Bewegung) `M64 P3` (sofort)
Enable (aktivieren) Brenner (engl. torch)	`M63 P3` (synchronisiert mit der Bewegung) `M65 P3` (sofort)
Setzen der Geschwindigkeit auf einen Prozentsatz der Vorschubgeschwindigkeit.	`` `M67 E3 Q`*n* (synchronisiert mit der Bewegung) `` `M68 E3 Q`*n* (sofort) *n* ist der einzustellende Prozentsatz 10 ist das Minimum, darunter wird auf 100% gesetzt 100 ist das Maximum, darüber wird auf 100% gesetzt **Es wird empfohlen, M68 E3 Q0 sowohl in der Präambel als auch in der Postambel zu verwenden.**
Schneidegerät compensation - links vom Pfad	`G41.1 D#<_hal[plasmac.kerf-width]>`
Cutter compensation - rechts vom Pfad	`G42.1 D#<_hal[plasmac.kerf-width]>`

Beschreibung	Code
Cutter compensation aus	G40 **Beachten Sie, dass M62 bis M68 ungültig sind, wenn die Schneidwerkzeugkompensation aktiviert ist.**
Schneiden holes mit 60% Vorschub	#<holes> = 1 für Löcher mit einem Durchmesser von weniger als 32 mm (1.26")
Schneiden Sie holes mit 60% Vorschub, schalten Sie den Brenner am Ende der Bohrung aus, fahren Sie mit der Bohrung für den Überschnitt fort.	#<holes> = 2 für Löcher mit weniger als 32 mm (1,26") Durchmesser Überschnittlänge = 4 mm (0.157")
Schneiden von holes und Bögen mit 60% Vorschub.	#<holes> = 3 für Löcher mit weniger als 32 mm (1,26") Durchmesser für Bögen mit weniger als 16 mm (0,63") Radius
Schneiden Sie Löcher (engl. holes) und Bögen mit 60% Vorschub, schalten Sie den Brenner am Ende der Bohrung aus, setzen Sie den Bohrungsweg für den Überschnitt fort.	#<holes> = 4 für Löcher mit weniger als 32 mm (1,26") Durchmesser für Bögen kleiner als 16 mm (0.63") Radius Überschnittlänge = 4 mm (0.157")
Geben Sie den hole Durchmesser an für #<holes> = 1-4.	#<h_diameter> = *n* (*n* ist der Durchmesser, verwenden Sie das gleiche Einheitensystem wie der Rest der G-Code-Datei)
Spezifizieren der Geschwindigkeit für Löcher #<holes> = 1-4.	#<h_velocity> = *n* (*n* ist der Prozentsatz, stellen Sie den Prozentsatz der aktuellen Vorschubgeschwindigkeit)
Angabe der Länge des Überschnitts (engl. over cut).	#<oclength> = *n* (*n* ist die Länge, verwenden Sie das gleiche Einheitensystem wie der Rest der G-Code-Datei)
Festlegen des Modus pierce-only.	#<pierce-only> = *n* (*n* ist der Modus, 0=normaler Schnittmodus, 1=nur Durchstichmodus)
Materialien erstellen oder bearbeiten. Optionen: 0 - Temporäre Vorgabe erstellen 1 - Hinzufügen, wenn nicht vorhanden 2 - Überschreiben, wenn vorhanden, sonst neu hinzufügen	obligatorische Parameter: (o=<*option*>, nu=<*nn*>, na=<*ll*>, ph=<*nn*>, pd=<*nn*>, ch=<*nn*>, fr=<*nn*>) optionale Parameter: (kw=<*nn*>, th=<*nn*>, ca=<*nn*>, cv=<*nn*>, pe=<*nn*>, gp=<*nn*>, cm=<*nn*>, jh=<*nn*>, jd=<*nn*>)

Beschreibung	Code
Z-Bewegung beibehalten	`#<keep-z-motion> = 1`

QtPlasmaC G-Code Beispiele

Beschreibung	Beispiel

Material auswählen und einen normalen Schnitt machen	`M190 P3`
	`M66 P3 L3 Q1`
	`F#<_hal[plasmac.cut-feed-rate]>`
	`M3 $0 S1`
	`.`
	`.`
	`M5 $0`

Geschwindigkeit auf 100% der CutFeedRate setzen	`M67 E3 Q0 or M67 E3 Q100`
Geschwindigkeit auf 60% der CutFeedRate setzen	`M67 E3 Q60`
Geschwindigkeit auf 40% der CutFeedRate setzen	`M67 E3 Q40`

Schneiden Sie ein Loch mit 60% reduzierter Geschwindigkeit mit der Geschwindigkeitseinstellung	`G21 (metric)`
	`G64 P0.05`
	`M52 P1 (adaptiven Vorschub aktivieren)`
	`F#<_hal[plasmac.cut-feed-rate]>`
	`G0 X10 Y10`
	`M3 $0 S1 (start cut)`
	`G1 X0`
	`M67 E3 Q60 (reduce feed rate to 60%)`
	`G3 I10 (the hole)`
	`M67 E3 Q100 (restore feed rate to 100%)`
	`M5 $0 (end cut)`
	`G0 X0 Y0`
	`M2 (end job)`

Schneiden Sie ein Loch mit 60% reduzierter Geschwindigkeit mit dem Befehl #<holes>	`G21 (metric)`
	`G64 P0.05`
	`M52 P1 (adaptiven Vorschub aktivieren)`
	`#<holes> = 1 (velocity reduction for holes)`
	`F#<_hal[plasmac.cut-feed-rate]>`
	`G0 X10 Y10`
	`M3 $0 S1 (start cut)`
	`G1 X0`
	`G3 I10 (the hole)`
	`M5 $0 (end cut)`
	`G0 X0 Y0`
	`M2 (end job)`

Beschreibung	Beispiel
Schneiden Sie ein Loch mit Überschnitt mit Brenner deaktivieren	G21 (metric) G64 P0.05 M52 P1 (adaptiven Vorschub aktivieren) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 M67 E3 Q60 (reduce feed rate to 60%) G3 I10 (the hole) M62 P3 (turn torch off) G3 X0.8 Y6.081 I10 (continue motion for 4 mm) M63 P3 (allow torch to be turned on) M67 E3 Q0 (restore feed rate to 100%) M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Schneiden Sie ein Loch mit Überschnitt mit dem Befehl #<holes>	G21 (metric) G64 P0.05 M52 P1 (adaptiven Vorschub aktivieren) #<holes> = 2 (over cut for holes) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 G3 I10 (the hole) M5 $0 (end cut) G0 X0 Y0 M2 (end job)
Schneiden Sie ein Loch mit 6,5 mm Überschnitt mit dem Befehl #<holes>	G21 (metric) G64 P0.05 M52 P1 (adaptiven Vorschub aktivieren) #<holes> = 2 (over cut for holes) #<oclength> = 6.5 (6.5 mm over cut length) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (start cut) G1 X0 G3 I10 (the hole) M5 $0 (end cut) G0 X0 Y0 M2 (end job)

Beschreibung	Beispiel

Wählen Sie Ritzer/Schreiber (engl. scribe) und wählen Sie den Brenner am Ende des Ritzens	`.`
	`.`
	`M52 P1 (adaptiven Vorschub aktivieren)`
	`F#<_hal[plasmac.cut-feed-rate]>`
	`T1 M6 (select scribe)`
	`G43 H0 (apply offsets)`
	`M3 $1 S1 (start plasmac with scribe)`
	`.`
	`.`
	`T0 M6 (select torch)`
	`G43 H0 (apply offsets)`
	`G0 X0 Y0 (parking position)`
	`M5 $1 (end)`

Lochmitte Spotting.	`(Erfordert einen kleinen Bewegungsbefehl, sonst passiert nichts)`
	`G21 (metrisch)`
	`F99999 (hohe Vorschubgeschwindigkeit)`
	`G0 X10 Y10`
	`M3 $2 S1 (spotting ein)`
	`G91 (relativer Abstandsmodus)`
	`G1 X0.000001`
	`G90 (absoluter Abstandsmodus)`
	`M5 $2 (spotting aus)`
	`G0 X0 Y0`
	`G90`
	`M2`

Temporäres Standardmaterial erstellen	`(o=0, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000)`
Material bearbeiten, falls nicht vorhanden, ein neues Material anlegen	`(o=2, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, kw=1.0)`

Mesa THCAD

Das Mesa THCAD ist eine gängige Methode, um die Lichtbogenspannung von einem Plasmaschneider zu erhalten, und ist auch für die ohmsche Messung des Materials während des Abtastens nützlich. Das THCAD kann sowohl für Konfigurationen mit parallelen Anschlüssen als auch für Konfigurationen mit Mesa Electronics-Hardware verwendet werden. Das THCAD ist in drei verschiedenen Modellen erhältlich: THCAD-5, THCAD-10 und THCAD-300.

Auf jeder THCAD-Karte befindet sich ein Mode-Jumper, der auf **UNIPOLAR** gesetzt werden sollte.

Auf jeder THCAD-Karte befindet sich ein Frequenzteiler-Jumper, der je nach Hardware-Typ gesetzt werden sollte:

Eingabegerät (engl. input device)	Empfohlene Einstellung
Parallele Schnittstelle mit sehr geringer Latenzzeit	F/32
Parallele Schnittstelle empfohlener Startpunkt	F/64
Paralleler Anschluss mit höherer Latenzzeit oder beim Schneiden von dickem Material	F/128
Mesa Karte	F/32

Dieser Wert muss während der Installation in PnCConf eingegeben werden.

NOTE Bei Verwendung einer parallelen Schnittstelle kann es erforderlich sein, dass der Benutzer die Jumper-Einstellung und die nachfolgenden Skalierungswerte auf der Registerkarte Parameters Tab anpasst, um optimale Ergebnisse zu erzielen. Zu den Symptomen gehören zufällige Brenneranstiege oder -absenkungen während des ansonsten stabilen Schneidens. Halscope-Diagramme können bei der Diagnose dieser Probleme hilfreich sein.

Auf der Rückseite des THCADs befindet sich ein Kalibrierungsaufkleber:

```
   THCAD-nnn

0V  121.1 kHz
5V  925.3 kHz
```

oder ähnliche Werte, müssen diese Werte bei der Installation in PnCConf eingegeben werden.

PnCConf hat Einträge für alle erforderlichen THCAD-Parameter und berechnet und konfiguriert alle erforderlichen Einstellungen. Die verwendeten Berechnungen sind wie folgt:

Spannungsskala

```
vs = r / ((f - z) / d / v)
```

Spannungs-Offset

```
vo = z / d
```

r = Teilerverhältnis (siehe unten).

f = Skalenendwert vom Kalibrierungsaufkleber.

z = 0 V-Wert vom Kalibrierungsaufkleber.

d = Wert von Jumper oben.

v = volle Skalenspannung von THCAD

Teiler-Verhältnis

THCAD-5 oder THCAD-10

Bei Anschluss an einen Plasma-CNC-Anschluss wird das Teilerverhältnis von der Plasmamaschine gewählt. Ein häufig verwendetes Verhältnis ist 20:1.

Beim Anschluss an die volle Lichtbogenspannung der Plasmamaschine wird für ein THCAD-10 in der Regel ein 1 MΩ-Widerstand vom negativen Lichtbogen zum negativen THCAD und ein 1 MΩ-Widerstand vom positiven Lichtbogen zum positiven THCAD verwendet. Das Teilerverhältnis ergibt sich aus:

```
r = (total_resistance + 100000) / 100000
```

THCAD-300

```
r = 1
```

IMPORTANT WENN DER BENUTZER EINE HF-STARTPLASMA-STROMVERSORGUNG VERWENDET, SOLLTE JEDER DIESER WIDERSTÄNDE AUS MEHREREN HOCHSPANNUNGSWIDERSTÄNDEN BESTEHEN.

CAUTION WENN DER BENUTZER EINE HF-STARTPLASMA-STROMVERSORGUNG VERWENDET, WIRD EINE OHMSCHE ABTASTUNG NICHT EMPFOHLEN.

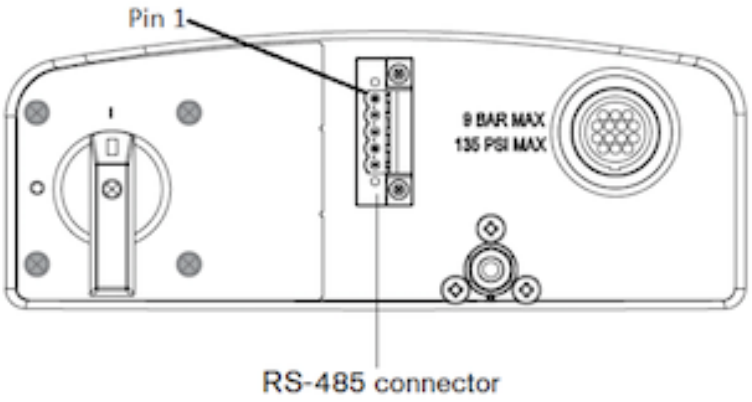
NOTE Diese Werte können mit diesem Online-Rechner berechnet werden.

NOTE Es gibt einen Tiefpass-Filter, der nützlich sein kann, wenn ein THCAD verwendet wird und die zurückgegebene Lichtbogenspannung stark verrauscht ist.

RS485-Verbindungen

Hypertherm RS485 Verdrahtungsplan (Drahtfarben innerhalb des Hypertherm in Klammern):

Anschluss am Maschinenpin #	Anschluss am Breakout Board
1 - Tx+ (Rot)	→ RXD+
2 - Tx- (schwarz)	→ RXD-
3 - Rx+ (braun)	→ T/R+
4 - Rx- (weiß)	→ T/R-
5 - GND (grün)	→ GND



RS485-Schnittstellen, von denen bekannt ist, dass sie funktionieren:

DTECH DT-5019 USB zu RS-485 Konverter Adapter:

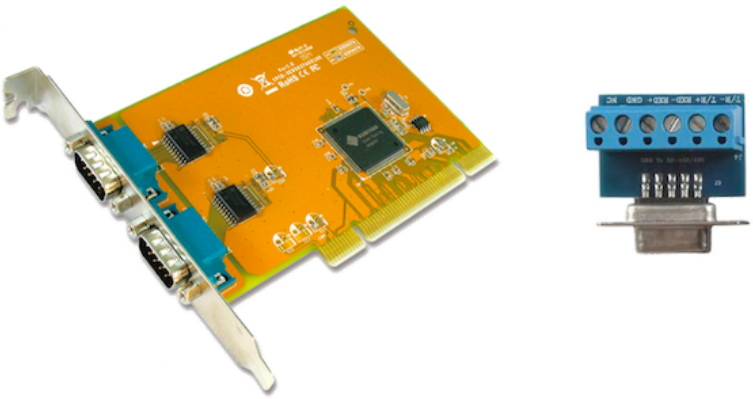


Um eine serielle Verbindung der Hauptplatine oder eine serielle Karte (RS232) in RS485 umzuwandeln, sind folgende Schritte erforderlich:

DTECH RS-232 zu RS-485 Konverter:



Beispiel einer seriellen Karte (Sunnix SER5037A PCI-Karte mit Breakout Board):

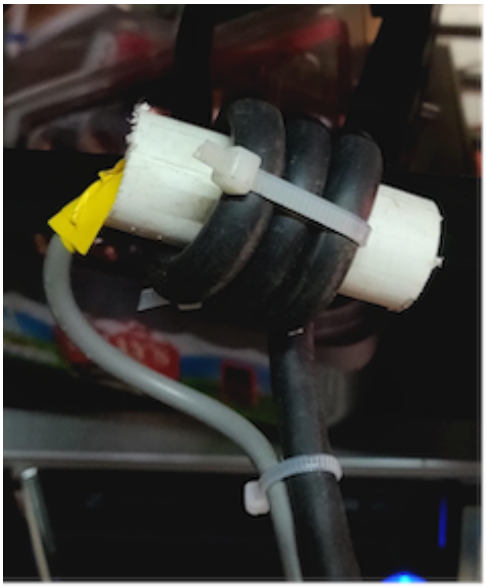


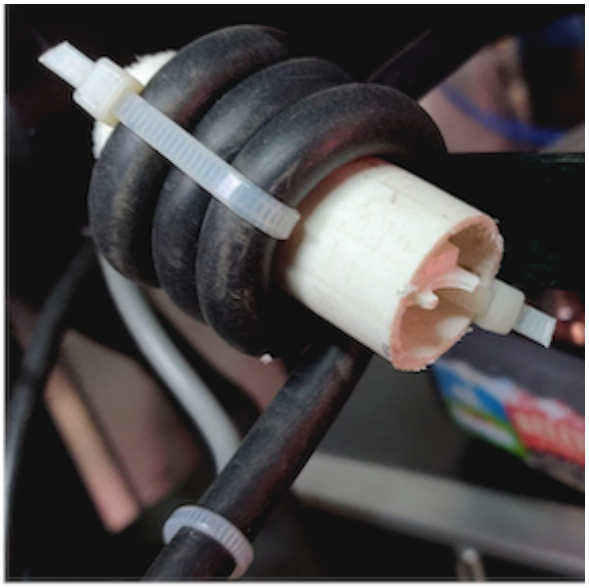
Lichtbogen OK mit einem Reed-Relais

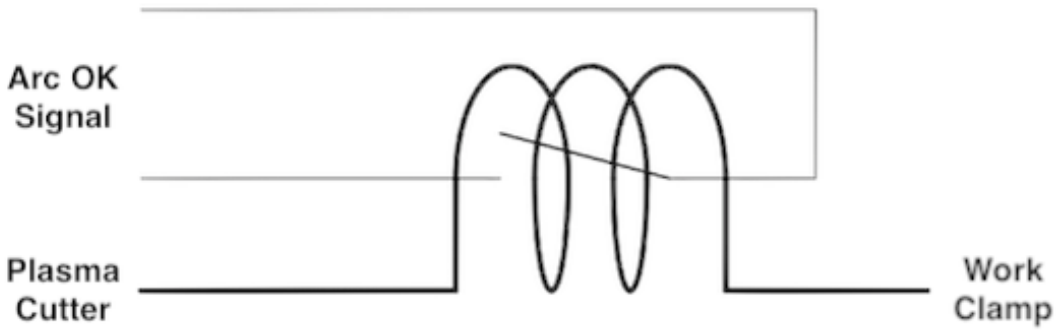
Eine effektive und sehr zuverlässige Methode, um ein Lichtbogen-OK-Signal von einer Plasmaversorgung ohne CNC-Anschluss zu erhalten, besteht darin, ein Reed-Relais in einer nicht leitenden Röhre zu montieren und drei Windungen des Arbeitskabels um die Röhre zu wickeln und zu sichern.

Diese Baugruppe fungiert nun als Relais, das sich einschaltet, wenn Strom durch die Arbeitsleitung fließt, was nur dann der Fall ist, wenn sich ein Lichtbogen gebildet hat.

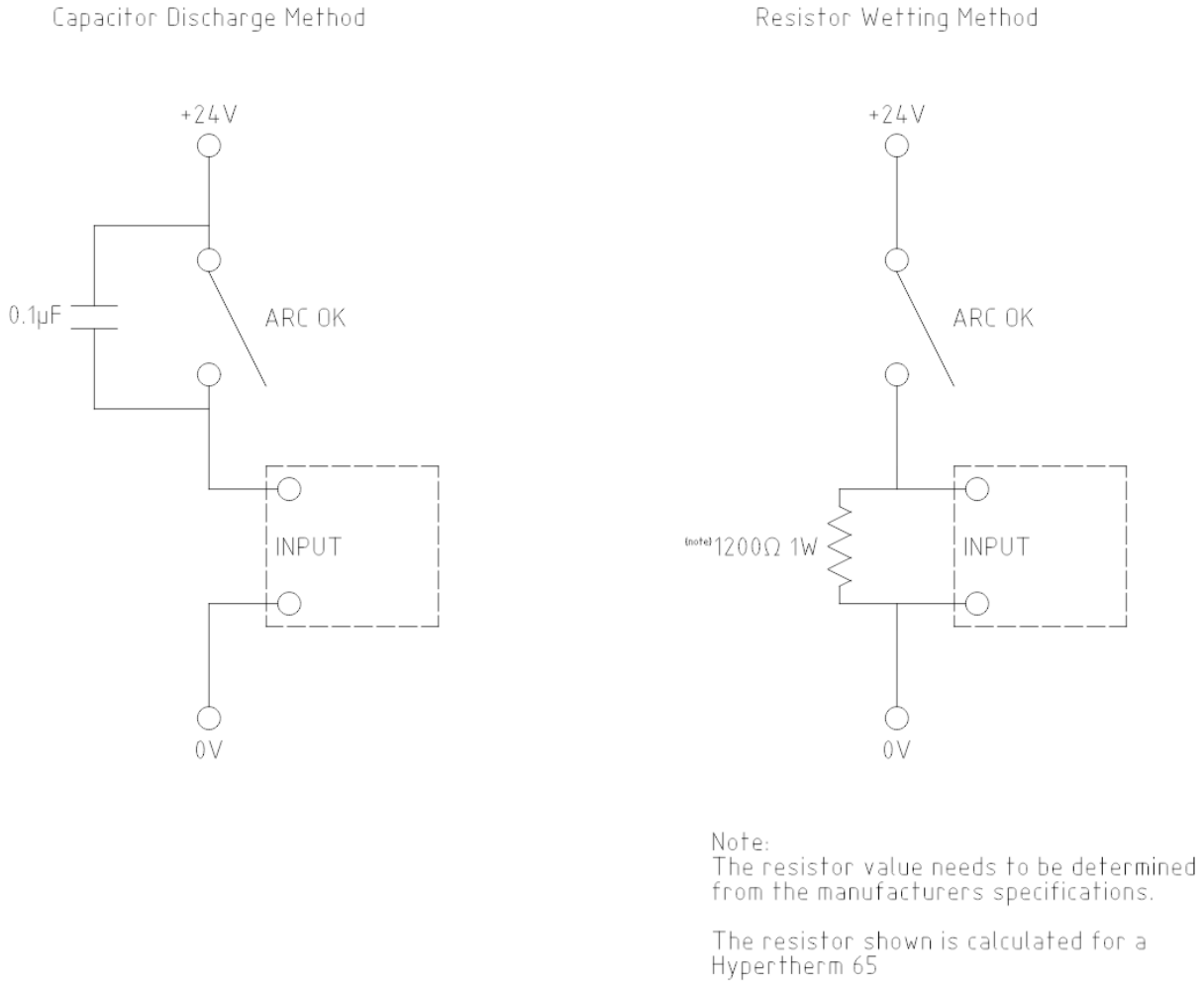
Dies erfordert, dass QtPlasmaC im Modus 1 und nicht im Modus 0 betrieben wird. Siehe die QtPlasmaC Modes Abschnitte für weitere Informationen.







Schematische Darstellung der Kontaktbelastung



Eine vollständige Beschreibung finden Sie unter Contact Load.

10.8.18. Bekannte Probleme

Tastatur-Jogging

Es gibt ein bekanntes Problem mit einigen Kombinationen von Hardware und Tastaturen, das die Autorepeat-Funktion der Tastatur beeinträchtigen kann und sich dann auf das Joggen der Tastatur auswirkt, indem es während des Joggens zeitweise stoppt und startet. Dieses Problem kann verhindert werden, indem die Autorepeat-Funktion des Betriebssystems für alle Tasten deaktiviert wird. QtPlasmaC verwendet diese Deaktivierung standardmäßig für alle Tasten nur, wenn das MAIN Tab sichtbar ist, mit den folgenden Ausnahmen, wenn Autorepeat bei sichtbarer MAIN Registerkarte erlaubt ist: G-Code-Editor ist aktiv, MDI ist aktiv. Wenn QtPlasmaC heruntergefahren wird, dann wird die Autorepeat-Funktion des Betriebssystems für alle Tasten aktiviert.

Wenn der Benutzer verhindern möchte, dass QtPlasmaC die Autorepeat-Einstellungen des Betriebssystems ändert, geben Sie die folgende Option in den **[GUI_OPTIONS]** Abschnitt der Datei <*Maschinenname*>.prefs ein:

```
Autorepeat all == True
```

Dieses Problem betrifft nicht das Joggen mit den GUI-Jog-Tasten.

NOTE Das Trennen und erneute Anschließen einer Tastatur während einer aktiven QtPlasmaC-Sitzung führt dazu, dass sich die Autorepeat-Funktion automatisch wieder aktiviert, was zu einem unregelmäßigen Anhalten und Starten während des Joggens führen kann. Der Benutzer muss QtPlasmaC neu starten, um die Autorepeat-Funktion wieder zu deaktivieren.

NO_FORCE_HOMING

QtPlasmaC hält sich derzeit nicht an die folgende Angabe in der <*Maschinenname*>.ini-Datei:

```
NO_FORCE_HOMING = 1
```

Unabhängig von dieser Einstellung erfordert QtPlasmaC, dass die Maschine vor der Ausführung von MDI-Befehlen oder dem Start von Programmen homed sein muss.

10.8.19. Beitragen zur Entwickung von QtPlasmaC

Bugfixes und Verbesserungen an QtPlasmaC sind immer willkommen. Die bevorzugte Methode, Code beizutragen, besteht darin, eine Pull-Anforderung (engl. pull request, PR) für einen einzigen Commit zum LinuxCNC GitHub-Repository einzureichen. Weitere Informationen zur Erstellung eines PR finden Sie in der LinuxCNC Dokumentation. Die einzige Voraussetzung ist, dass Sie ein GitHub Konto anlegen. Alle PRs werden überprüft und dann von einem der Entwickler akzeptiert. Wenn Sie mit der Einreichung eines PR noch nicht vertraut sind oder sich unwohl fühlen, so können Sie alternativ die Codeänderungen in einem LinuxCNC Forum Thread vorstellen.

Bugfixes werden sowohl für den neusten freigegebenen Entwicklungszweig (engl. und beinahe eingedeutscht: branch) als auch für den master branch akzeptiert. Wenn ein Bugfix für beide branches gilt, so ist nur der PR für den branch der neuesten release einzureichen, da sie von einem Entwickler für den master branch angepasst wird.

Verbesserungen werden nur für den Hauptzweig akzeptiert.

Jede PR, mit Ausnahme von Änderungen der QtPlasmaC-Dokumentation, erfordert, dass die entsprechende Versionsnummer erhöht wird und auch die Versionshistorie aktualisiert wird. Die Versionsnummern befinden sich an folgenden Stellen:

Lokalisation	Format	Erhöht wenn
src/hal/components/plasmac.comp	nnn	Komponenten Code Änderungen
share/qtvcp/screens/qtplasmac/qtplasmac_handler.py	nnn.*nnn*	Komponenten Code-Änderungen *GUI Code Änderungen*

Die Versionshistorie befindet sich bei share/qtvcp/screens/qtplasmac/versions.html.

10.8.20. Unterstützung

Online-Hilfe und -Unterstützung finden Sie unter PlasmaC section des LinuxCNC Forum.

Der Benutzer kann eine komprimierte Datei erstellen, welche die komplette Maschinenkonfiguration enthält, um die Fehlerdiagnose zu unterstützen, indem er den Anweisungen im Abschnitt backup folgt. Die resultierende Datei eignet sich zum Anhängen an einen Beitrag auf dem LinuxCNC Forum, um der Gemeinschaft zu helfen, bestimmte Probleme zu diagnostizieren.

10.9. MDRO GUI

10.9.1. Einführung

MDRO ist eine einfache grafische Front-End für LinuxCNC bietet eine Anzeige von Daten aus Digital Read Out (DRO) Skalen. Es bietet Funktionalität ähnlich wie ein normaler Maschinist DRO-Anzeige, so dass der Benutzer die DRO-Skalen auf der Maschine zu verwenden, wenn der Betrieb in einem manuellen-only (Hand-Kurbel) Modus. Sie ist besonders nützlich für manuelle Maschinen, wie z. B. mit DRO ausgestattete Bridgeport-Fräsmaschinen, die auf CNC umgerüstet wurden, aber noch über manuelle Bedienelemente verfügen.

MDRO ist Maus- und Touchscreen-freundlich.

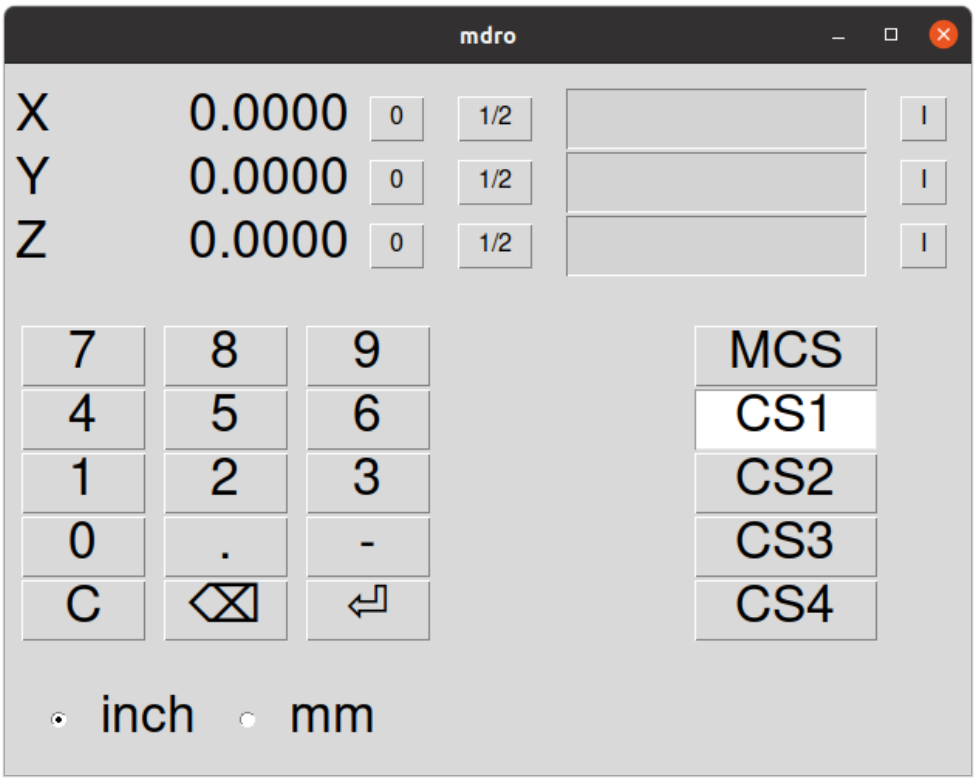


Figure 198. MDRO Fenster

10.9.2. Erste Schritte

Wenn Ihre Konfiguration derzeit nicht für die Verwendung von MDRO eingerichtet ist, können Sie dies durch Bearbeiten der INI-Datei ändern. Ändern Sie im Abschnitt `[DISPLAY]` die Zeile `DISPLAY =` in

`DISPLAY = mdro`. MDRO ist standardmäßig auf XYZ für die Achsen eingestellt, aber das kann geändert werden. Setzen Sie den Abschnitt `[DISPLAY] ` auf `GEOMETRY = XYZ` für eine 3-Achsen-Fräse. Bei einer Drehmaschine mit DRO-Skalen für die X- und Z-Achse könnte `GEOMETRY = XZ` verwendet werden.

Wenn MDRO gestartet wird, öffnet sich ein Fenster wie das in der Abbildung MDRO Fenster oben.

INI-Datei Optionen

Weitere Optionen, die im Abschnitt "[DISPLAY]" enthalten sein können, sind:

- `MDRO_VAR_FILE = <file.var>` - Vorladen von G54 - G57 Koordinatensystemdaten.
  - Vorladen einer .var-Datei. Dies ist in der Regel die vom operativen Code verwendete .var-Datei.
- `POINT_SIZE = <n>` - Setzt die Punktgröße des Textes.
  - Mit dieser Option wird die Größe der verwendeten Schrift festgelegt, wodurch sich auch die Gesamtgröße des Fensters ergibt. Die Standardschriftgröße ist 20, typische Größen sind 20 bis 30.
- `MM = 1` Stellen Sie dies ein, wenn die DRO-Skalen in Millimeter skalierte Daten liefern.

Kommandozeilen-Optionen

MDRO kann mit dem Befehl `loadusr` in einer HAL-Datei gestartet werden. Optionen, die denen in der INI-Datei entsprechen, können in der Befehlszeile gesetzt werden:

- *-l <file.var>* - Daten des G54 - G57-Koordinatensystems vorladen.
- `-p <n>` - Setzt die Punktgröße des Textes.
- `-m` - Stellen Sie dies ein, wenn die DRO-Skalen in Millimeter skalierte Daten liefern.
- `<axes>` - anzuzeigende Achsen. Siehe „GEOMETRIE“ oben.

Pins

Bei einem Beispiel mit "XYZA" als AXES-Argument werden diese Pins beim Start von MDRO erstellt:

```
mdro.axis.0
mdro.axis.1
mdro.axis.2
mdro.axis.3
mdro.index-enable.0
mdro.index-enable.1
mdro.index-enable.2
mdro.index-enable.3
```

In diesem Beispiel ist die erste Zeile der Anzeige mit "X" beschriftet und zeigt die Daten der DRO-Skala, die an den Pin "mdro.axis.0" angeschlossen ist. Die Pins `mdro.index-enable.n` sollten mit den Index-Pins des DRO verbunden werden, wenn das DRO sie unterstützt.

Die Pins müssen in der im Eintrag `POSTGUI_HALFILE` der INI-Datei angegebenen Datei angeschlossen werden, wenn das Programm aus einer INI-Datei gestartet wird. Sie können direkt nach dem Befehl

loadusr gesetzt werden, wenn das Programm in einer HAL-Datei gestartet wird.

10.9.3. MDRO Fenster

Das MDRO-Fenster enthält die folgenden Elemente:

- Eine Zeile für jede Achse. Jede Zeile enthält:
  - der Name der Achse,
  - der aktuelle Wert,
  - ein "z"-Button, der den Wert auf Null setzt
  - ein Button "1/2", der den Wert halbiert
  - ein Eingabefeld, in dem ein benutzerdefinierter Wert eingegeben werden kann. Dieses Feld kann über die Tastatur oder über das Bildschirmtastenfeld eingestellt werden.
  - Ein "I"-Button, der einen Indexvorgang startet (siehe unten),
- ein Tastenfeld, mit dem über eine Maus oder einen Touchscreen Werte in das Eingabefeld eingegeben werden können,
- Koordinatensystem Auswahl Buttons:
  - Mit dem Button "mcs" wird das Maschinenkoordinatensystem ausgewählt. Dies sind die Rohwerte der an die Pins `mdro.axis.__n__` angeschlossenen Messgeräte.
  - Mit den Buttons "cs1" - "cs4" kann der Benutzer eines von vier benutzerdefinierten Koordinatensystemen auswählen. Wenn das Programm mit der Option `MDRO_VAR_FILE =` gestartet wird, werden die Beschriftungen in "g54" - "g57" geändert und die Werte aus der angegebenen .var-Datei werden vorgeladen. Beachten Sie, dass alle Änderungen an den Werten nicht dauerhaft sind: Die .var-Datei wird nie geändert.
- Inch/Millimeter-Auswahltasten.

10.9.4. Index-Operationen

`MDRO` unterstützt DRO-Skalen mit Indexmarken. Klicken Sie auf die Schaltfläche "I" in der Achsenzeile und kurbeln Sie die Achse auf die Indexposition. Die Maschinenkoordinate wird auf Null gesetzt. Dies ist am einfachsten beim Start oder bei Auswahl des Koordinatensystems "mcs" zu erkennen.

10.9.5. Simulation

Der einfachste Weg zu sehen, wie `MDRO` funktioniert, ist, es in einer Simulationsumgebung auszuprobieren. Fügen Sie diesen Abschnitt an das Ende Ihrer Simulations-HAL-Datei an, normalerweise "hallib/core_sim.hal":

```
loadusr -W mdro -l sim.var XYZ
net x-pos-fb  => mdro.axis.0
net y-pos-fb  => mdro.axis.1
net z-pos-fb  => mdro.axis.2
```

[1] In den USA wird der Buchstabe V üblicherweise als Symbol (Voltage) und als Einheit (Volt) verwendet.

Chapter 11. G-Code Programmierung

11.1. Koordinatensysteme

11.1.1. Einführung

In diesem Kapitel werden wir versuchen, Koordinatensysteme zu entmystifizieren. Es ist ein sehr wichtiges Konzept, um den Betrieb einer CNC-Maschine, ihre Konfiguration und ihre Verwendung zu verstehen.

Wir werden auch zeigen, dass es sehr interessant ist, einen Referenzpunkt auf dem Rohling oder dem Werkstück zu verwenden und das Programm von diesem Punkt aus arbeiten zu lassen, ohne zu berücksichtigen, wo das Werkstück auf dem Tisch liegt.

Dieses Kapitel führt Sie ein in die Beschreibung von Verschiebungen ein, wie sie von LinuxCNC verwendet werden. Je nach Kontext möchte man auch Versatz sagen, oder Kompensation oder aus dem Englischen eingedeutscht auch gern Offsets (buchstäblich: danebengesetzt) beibehalten. Dazu gehören:

- Maschinenkoordinaten (G53)
- Neun Koordinatensystem-Offsets (G54-G59.3)
- Globale Offsets (G92) und lokale Offsets (G52)

11.1.2. Maschinenkoordinatensystem

Beim Start von LinuxCNC ist jeweilige Positionen der einzelnen Achsen auch der Ursprung der Maschine. Sobald eine Achse referenziert ist, wird der Maschinenursprung für diese Achse auf die referenzierte Position gesetzt. Der Maschinenursprung ist das Maschinenkoordinatensystem, auf dem alle anderen Koordinatensysteme basieren. Der G-Code G53 kann verwendet werden, um sich im Maschinenkoordinatensystem zu bewegen.

Maschinenkoordinaten bewegen sich: G53

Unabhängig von einem eventuell aktiven Offset weist ein G53 in einer Codezeile den Interpreter an, die angegebenen tatsächlichen Achsenpositionen (absolute Positionen) anzufahren. Zum Beispiel:

```
G53 G0 X0 Y0 Z0
```

fährt von der aktuellen Position zu der Position, an der die Maschinenkoordinaten der drei Achsen auf Null stehen. Sie können diesen Befehl verwenden, wenn Sie eine feste Position für den Werkzeugwechsel haben oder wenn Ihre Maschine über einen automatischen Werkzeugwechsler verfügt. Sie können diesen Befehl auch verwenden, um den Arbeitsbereich zu räumen und auf das Werkstück im Schraubstock zuzugreifen.

G53 ist ein nicht modaler Befehl. Er muss in jedem Satz verwendet werden, in dem eine Bewegung im Maschinenkoordinatensystem gewünscht ist.

11.1.3. Koordinatensysteme

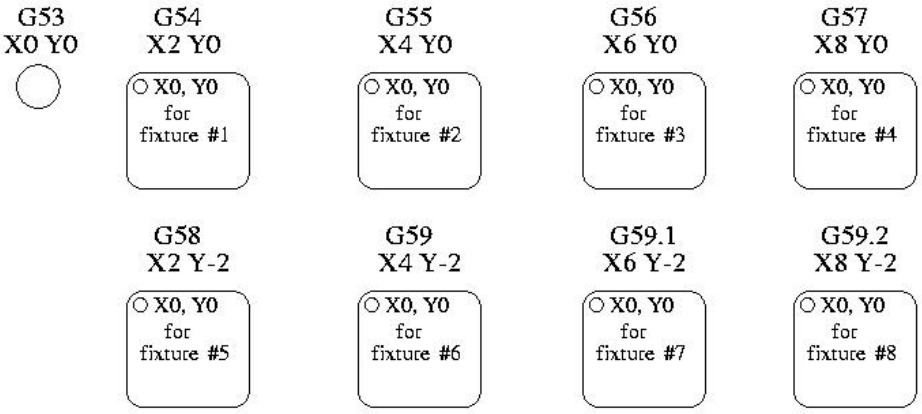


Figure 199. Beispiel für Koordinatensysteme

Koordinatensystem-Offsets

- G54 - Koordinatensystem 1 verwenden
- G55 - Koordinatensystem 2 verwenden
- G56 - Koordinatensystem 3 verwenden
- G57 - Koordinatensystem 4 verwenden
- G58 - Koordinatensystem 5 verwenden
- G59 - Koordinatensystem 6 verwenden
- G59.1 - Koordinatensystem 7 verwenden
- G59.2 - Koordinatensystem 8 verwenden
- G59.3 - Koordinatensystem 9 verwenden

Koordinatensystem-Offsets werden verwendet, um das Koordinatensystem gegenüber dem Maschinenkoordinatensystem zu verschieben. Dadurch kann der G-Code für das Werkstück unabhängig von der Position des Werkstücks auf der Maschine programmiert werden. Die Verwendung von Koordinatensystem-Offsets würde es Ihnen ermöglichen, Teile an mehreren Stellen mit demselben G-Code zu bearbeiten.

Die Werte für die Offsets sind in der VAR-Datei, die von der INI-Datei während des Starts eines LinuxCNC angefordert wird gespeichert. Im folgenden Beispiel, das G55 verwendet, wird die Position jeder Achse für G55 Ursprung in einer nummerierten Variablen gespeichert.

Im VAR-Dateischema speichert die erste Variablennummer den X-Offset, die zweite den Y-Offset und so weiter für alle neun Achsen. Für jeden Offset des Koordinatensystems gibt es nummerierte Sätze dieser Art.

Jede der grafischen Oberflächen verfügt über eine Möglichkeit, Werte für diese Offsets festzulegen. Sie können diese Werte auch festlegen, indem Sie die VAR-Datei selbst bearbeiten und dann LinuxCNC neu starten, so dass die LinuxCNC die neuen Werte liest, dies jedoch nicht der empfohlene Weg ist. Die Verwendung von G10, G52, G92, G28.1 usw. sind bessere Möglichkeiten, die Variablen festzulegen. In unserem Beispiel bearbeiten wir die Datei direkt, sodass G55 die folgenden Werte annimmt:

Table 80. Beispiel für G55-Parameter

Achse	Variable	Wert
X	5241	2.000000
Y	5242	1.000000
Z	5243	-2.000000
A	5244	0.000000
B	5245	0.000000
C	5246	0.000000
U	5247	0.000000
V	5248	0.000000
W	5249	0.000000

Dies bedeutet, dass die Nullpositionen von G55 auf X = 2 Einheiten, Y = 1 Einheit und Z = -2 Einheiten von der absoluten Nullposition entfernt sind.

Sobald die Werte zugewiesen sind, würde ein Aufruf von G55 in einem Programmsatz den Nullbezug um die gespeicherten Werte verschieben. Die folgende Zeile würde dann jede Achse auf die neue Nullposition fahren. Im Gegensatz zu G53 sind G54 bis G59.3 modale Befehle. Sie wirken auf alle Codesätze, nachdem einer von ihnen gesetzt wurde. Das Programm, das unter Verwendung von Vorrichtungsoffsets ausgeführt werden könnte, würde nur eine einzige Koordinatenreferenz für jede der Positionen und alle dort auszuführenden Arbeiten erfordern. Der folgende Code ist ein Beispiel für die Herstellung eines Quadrats unter Verwendung der G55-Offsets, die wir oben festgelegt haben.

```
G55 ; Nutze Koordinaten-System 2
G0 X0 Y0 Z0
G1 F2 Z-0.2000
X1
Y1
X0
Y0
G0 Z0
G54 ; Nutze coordinaten-System 1
G0 X0 Y0 Z0
M2
```

In diesem Beispiel verlässt der G54 gegen Ende das G54-Koordinatensystem mit allen Nullpunktverschiebungen, so dass es einen Modalcode für die absoluten maschinenbasierten Achsenpositionen gibt. Dieses Programm geht davon aus, dass wir dies getan haben und verwendet den Endbefehl als einen Befehl zum Maschinennullpunkt. Es wäre möglich gewesen, G53 zu verwenden und an dieselbe Stelle zu gelangen, aber dieser Befehl wäre nicht modal gewesen, und alle danach erteilten Befehle hätten wieder die G55-Offsets verwendet, da dieses Koordinatensystem noch in Kraft wäre.

[source,ngc]

```
G54 verwendet die Parameter des Koordinatensystems 1
G55 verwendet die Parameter des Koordinatensystems 2
G56 verwendet die Parameter des Koordinatensystems 3
G57 verwendet Parameter des Koordinatensystems 4
G58 verwendet Parameter des Koordinatensystems 5
G59 verwendet Parameter des Koordinatensystems 6
G59.1 verwendet Parameter des Koordinatensystems 7
G59.2 verwendet Parameter des Koordinatensystems 8
G59.3 verwendet die Parameter des Koordinatensystems 9
```

Standard-Koordinatensystem

Eine weitere Variable in der VAR-Datei wird wichtig, wenn wir über Offset-Systeme nachdenken. Diese Variable heißt 5220. In den Standarddateien ist ihr Wert auf 1.00000 gesetzt. Dies bedeutet, dass, wenn LinuxCNC startet das erste Koordinatensystem als Standard verwendet werden. Wenn Sie diesen Wert auf 9.00000 setzen, würde er das neunte Offset-System als Standard für das Starten und Zurücksetzen verwenden. Jeder andere Wert als eine ganze Zahl (dezimal wirklich) zwischen 1 und 9, oder eine fehlende 5220 Variable wird die LinuxCNC auf den Standardwert von 1.00000 beim Start zurückkehren.

Koordinatensystem-Offsets einstellen

Der Befehl G10 L2x kann verwendet werden, um Koordinatensystem-Offsets zu setzen:

- G10 L2 P(1-9)' - Offset(s) auf einen Wert setzen. Aktuelle Position irrelevant (siehe G10 L2 für Details).
- G10 L20 P(1-9)' - Offset(s) setzen, so dass die aktuelle Position zu einem Wert wird (siehe G10 L20 für Details).

NOTE Wir geben hier nur einen kurzen Überblick, eine vollständige Beschreibung finden Sie in den G-Code-Abschnitten.

11.1.4. Lokale und globale Offsets

Der Befehl G52

G52' wird in einem Teileprogramm als temporärer "lokaler Koordinatensystemversatz" innerhalb des Werkstückkoordinatensystems verwendet. Ein Beispiel für einen Anwendungsfall ist die Bearbeitung mehrerer identischer Features an verschiedenen Stellen eines Werkstücks. Für jedes Feature programmiert *G52* einen lokalen Referenzpunkt innerhalb der Werkstückkoordinaten, und ein Unterprogramm wird aufgerufen, um das Feature relativ zu diesem Punkt zu bearbeiten.

Die Achsversätze *G52* werden relativ zu den Werkstückkoordinatenversätzen *G54* bis *G59.3* programmiert. Als lokaler Versatz wird *G52* nach dem Werkstückversatz angewendet, einschließlich Drehung. Auf diese Weise wird ein Teilemerkmal auf jedem Teil identisch bearbeitet, unabhängig von der Ausrichtung des Teils auf der Palette.

CAUTION Als temporäre Offset, Set und Unset innerhalb der lokalisierten Umfang eines Teils Programm, in anderen G-Code-Interpreter *G52* nicht nach Maschinen-Reset, *M02*

	oder *M30* persistieren. In LinuxCNC, *G52* teilt Parameter mit *G92*, die, aus historischen Gründen, **persistieren**. Siehe G92 Persistence Cautions unten.
CAUTION	*G52* und *G92* teilen sich die gleichen Offset-Register. Daher überschreibt die Einstellung von *G52* jede frühere Einstellung von *G92*, und *G52* bleibt über das Zurücksetzen der Maschine hinaus erhalten, wenn die *G92*-Persistenz aktiviert ist. Diese Wechselwirkungen können zu unerwarteten Offsets führen. Siehe G92- und G52-Interaktionshinweise weiter unten.

Durch die Programmierung von *G52 X1 Y2* wird die X-Achse des aktuellen Werkstückkoordinatensystems um 1 und die Y-Achse um 2 verschoben. Dementsprechend werden die X- und Y-Koordinaten der aktuellen Werkzeugposition um 1 bzw. 2 verringert. Achsen, die im Befehl nicht festgelegt wurden, wie z. B. die Z-Achse im vorigen Beispiel, bleiben unberührt: Jede frühere *G52*-Z -Verschiebung bleibt wirksam, und andernfalls ist die Z-Verschiebung Null.

Der temporäre lokale Offset kann mit *G52 X0 Y0* gelöscht werden. Alle Achsen, die nicht explizit auf Null gesetzt wurden, behalten den vorherigen Offset bei.

G52 hat die gleichen Offset-Register wie *G92*, und daher ist *G52* auf der DRO und der Vorschau mit der Bezeichnung *G92* sichtbar.

11.1.5. G92-Achsen-Offsets

G92 ist der am meisten missverstandene und cleverste Befehl, der mit LinuxCNC programmierbar ist. Die Art und Weise, wie es funktioniert hat ein bisschen zwischen den ersten Versionen und der aktuellen geändert. Diese Änderungen haben zweifellos viele Benutzer verwirrt. Sie sollten als ein Befehl erzeugt eine temporäre Offset, die für alle anderen Offsets gilt gesehen werden.

Die G92-Befehle

G92 wird typischerweise auf zwei konzeptionell unterschiedliche Arten verwendet: als "globaler Koordinatensystem-Offset" oder als "lokaler Koordinatensystem-Offset".

Der *G92*-Befehlssatz umfasst:

- *G92* - Wenn dieser Befehl mit Achsennamen verwendet wird, werden Werte auf Offset-Variablen festgelegt.
- *G92.1* - Dieser Befehl setzt Nullwerte für die G92-Variablen.
- *G92.2* - Dieser Befehl setzt die G92-Variablen außer Kraft, setzt sie aber nicht auf Null.
- *G92.3* - Dieser Befehl wendet wieder Offset-Werte an, die zuvor ausgesetzt wurden.

Als globale Verschiebung wird *G92* verwendet, um alle Werkstückkoordinatensysteme *G54* bis *G59.3* zu verschieben. Ein Beispiel für einen Anwendungsfall ist die Bearbeitung mehrerer identischer Teile in Aufspannungen mit bekannten Positionen auf einer Palette, aber die Position der Palette kann sich zwischen Läufen oder zwischen Maschinen ändern. Jede Verschiebung der Aufspannvorrichtung in Bezug auf einen Referenzpunkt auf der Palette wird in einem der Werkstückkoordinatensysteme *G54* bis *G59.3* voreingestellt, und *G92* wird verwendet, um den Referenzpunkt auf der Palette "anzutasten". Dann

wird für jedes Teil das entsprechende Werkstückkoordinatensystem ausgewählt und das Teileprogramm ausgeführt.

NOTE Die Drehung des Werkstückkoordinatensystems *G10 R-* ist spezifisch für den Interpreter *rs274ngc*, und der Offset *G92* wird *nach* der Drehung angewendet. Wenn *G92* als globaler Offset verwendet wird, kann die Drehung des Werkstückkoordinatensystems zu unerwarteten Ergebnissen führen.

Als lokales Koordinatensystem wird *G92* als temporärer Versatz innerhalb des Werkstückkoordinatensystems verwendet. Ein Beispiel für einen Anwendungsfall ist die Bearbeitung eines Teils mit mehreren identischen Merkmalen an verschiedenen Stellen. Für jedes Feature wird *G92* verwendet, um einen lokalen Referenzpunkt zu setzen, und ein Unterprogramm wird aufgerufen, um das Feature ab diesem Punkt zu bearbeiten.

NOTE Von der Verwendung von *G92* wird bei der Programmierung mit lokalen Koordinatensystemen in einem Teileprogramm abgeraten. Siehe stattdessen *G52*, ein lokaler Koordinatensystem-Offset, der intuitiver ist, wenn der gewünschte Offset relativ zum Werkstück bekannt ist, aber die aktuelle Werkzeugposition möglicherweise nicht bekannt ist.

Die Programmierung von *G92 X0 Y0 Z0* setzt die aktuelle Werkzeugposition auf die Koordinaten X0, Y0 und Z0, ohne Bewegung. G92 arbeitet **nicht** mit absoluten Maschinenkoordinaten. Es arbeitet mit der **aktuellen Position**.

G92 funktioniert auch vom aktuellen Standort aus, der durch alle anderen Offsets geändert wird, die beim Aufruf des Befehls *G92* wirksam sind. Beim Testen auf Unterschiede zwischen Arbeitsversätzen und tatsächlichen Offsets wurde festgestellt, dass ein "G54"-Offset einen "G92" aufheben und somit den Anschein erwecken könnte, dass keine Offsets in Kraft waren. Die "G92" war jedoch immer noch für alle Koordinaten in Kraft und erzeugte erwartete Arbeitsversätze für die anderen Koordinatensysteme.

Standardmäßig werden die *G92*-Offsets nach dem Start der Maschine wiederhergestellt. Programmierer, die ein Fanuc-Verhalten wünschen, bei dem die *G92*-Offsets beim Maschinenstart und nach einem Reset oder Programmende gelöscht werden, können die *G92*-Persistenz deaktivieren, indem sie *DISABLE_G92_PERSISTENCE = 1* im Abschnitt *[RS274NGC]* der INI-Datei 'einstellen.

NOTE Es ist gute Praxis, die *G92* Offsets am Ende ihrer Verwendung mit *G92.1* oder *G92.2* zu löschen. Wenn Sie LinuxCNC mit aktivierter *G92*-Persistenz starten (die Voreinstellung), werden alle Offsets in den *G92*-Variablen angewendet, wenn eine Achse referenziert wird. Siehe G92 Persistenz Vorsichtsmaßnahmen unten.

G92 Werte festlegen

Es gibt mindestens zwei Möglichkeiten, G92-Werte festzulegen:

- Mit einem Rechtsklick auf die Positionsanzeigen in tklinuxcnc öffnet sich ein Fenster, in dem Sie einen Wert eingeben können.
- Mit dem Befehl G92

Beide gehen von der aktuellen Position der Achse aus, die verschoben werden soll.

Durch die Programmierung von *G92 X Y Z A B C U V W* werden die Werte der G92-Variablen so eingestellt, dass jede Achse den mit ihrem Namen verbundenen Wert annimmt. Diese Werte werden der aktuellen Position der Achsen zugewiesen. Diese Ergebnisse entsprechen den Absätzen eins und zwei des NIST-Dokuments.

G92-Befehle gehen von der aktuellen Achsenposition aus und addieren und subtrahieren korrekt, um der aktuellen Achsenposition den durch den G92-Befehl zugewiesenen Wert zu geben. Die Effekte funktionieren auch dann, wenn vorherige Offsets vorhanden sind.

Wenn also die X-Achse derzeit 2,0000 als Position anzeigt, wird mit *G92 X0* ein Offset von -2,0000 gesetzt, so dass die aktuelle Position von X Null wird. Ein *G92 X2* setzt einen Offset von 0.0000 und die angezeigte Position wird nicht verändert. Ein *G92 X5.0000* setzt einen Offset von 3.0000, so dass die aktuell angezeigte Position zu 5.0000 wird.

G92 Persistenz-Vorsichtsmaßnahmen

Standardmäßig werden die Werte eines *G92*-Offsets in der VAR-Datei gespeichert und nach einem Neustart der Maschine oder einem Neustart wiederhergestellt.

Die G92-Parameter sind:

- 5210 - Aktivieren/Deaktivieren der Flags (1.0/0.0)
- 5211 - Versatz (engl. offset) der X-Achse
- 5212 - Versatz der Y-Achse
- 5213 - Z-Achsen-Versatz
- 5214 - Versatz der A-Achse
- 5215 - Versatz der B-Achse
- 5216 - Versatz der C-Achse
- 5217 - Versatz der U-Achse
- 5218 - Versatz der V-Achse
- 5219 - Versatz der W-Achse

wobei 5210 das *G92*-Freigabeflag ist (1 für aktiviert, 0 für deaktiviert) und 5211 bis 5219 die Achsenoffsets sind. Wenn Sie unerwartete Positionen als Ergebnis einer befohlenen Bewegung sehen, weil Sie einen Offset in einem früheren Programm gespeichert und am Ende nicht gelöscht haben, geben Sie ein G92.1 im MDI-Fenster ein, um die gespeicherten Offsets zu löschen.

Wenn G92-Werte in der VAR-Datei vorhanden sind, wenn LinuxCNC startet, werden die G92-Werte in der Var-Datei auf die Werte der aktuellen Position jeder Achse angewendet werden. Wenn dies die Ausgangsposition ist und die Ausgangsposition als Maschinennullpunkt eingestellt ist, wird alles korrekt sein. Sobald die Ausgangsposition mit Hilfe von echten Maschinenschaltern oder durch Bewegen jeder Achse zu einer bekannten Ausgangsposition und Ausgeben eines Achsen-Ausgangsbefehls festgelegt wurde, werden alle G92-Offsets angewendet. Wenn Sie eine G92 X1 in Kraft haben und die X-Achse in

den Grundzustand bringen, wird die Positionsanzeige *X: 1.000* statt des erwarteten *X: 0.000* anzeigen, da die G92 auf den Maschinenursprung angewendet wurde. Wenn Sie ein G92.1 Befehl absetzen und die DRO zeigt nun überall Nullen, dann hatten Sie eine G92 Offset in aktiv als Sie zuletzt LinuxCNC ausführten.

Sofern Sie nicht die Absicht haben, dieselben G92-Offsets im nächsten Programm zu verwenden, ist es die beste Praxis, am Ende jeder G-Code-Datei, in der Sie G92-Offsets verwenden, einen G92.1 auszuführen.

Wenn ein Programm während der Verarbeitung abgebrochen wird, für das *G92*-Offsets gelten, werden diese beim Start wieder aktiv. Zur Sicherheit sollten Sie immer eine Präambel verwenden, um die Umgebung so einzustellen, wie Sie sie erwarten. Außerdem kann die *G92*-Persistenz durch Setzen von *DISABLE_G92_PERSISTENCE = 1* im Abschnitt *[RS274NGC]* der INI-Datei deaktiviert werden.

G92 und G52 Wechselwirkungen - Hinweise zur Vorsicht

G52 und *G92* teilen sich die gleichen Offset-Register. Sofern die *G92*-Persistenz in der INI-Datei nicht deaktiviert ist (siehe G92-Befehle), bleiben *G52*-Offsets auch nach dem Zurücksetzen der Maschine, *M02* oder *M30* bestehen. Beachten Sie, dass ein während eines Programmabbruchs wirksamer *G52*-Offset zu unbeabsichtigten Offsets führen kann, wenn das nächste Programm ausgeführt wird. Siehe obige G92 Warnungen zur Persistenz.

11.1.6. Beispielprogramme mit Offsets/Kompensationen

Beispielprogramm mit Werkstückkoordinaten-Versätzen

Dieses Beispielgravurprojekt fräst einen Satz von vier Kreisen mit einem Radius von 0,1, die sich in etwa sternförmig um einen zentralen Kreis herum befinden. Wir können die einzelnen Kreismuster wie folgt einrichten.

```
G10 L2 P1 X0 Y0 Z0 (sicherstellen, dass G54 auf Maschine Null eingestellt ist)
G0 X-0.1 Y0 Z0
G1 F1 Z-0,25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
M2
```

Wir können eine Reihe von Befehlen erteilen, um Versätze für die vier anderen Kreise wie folgt zu erstellen.

```
G10 L2 P2 X0.5 (verschiebt den G55 X-Wert um 0,5 Zoll)
G10 L2 P3 X-0.5 (verschiebt den G56 X-Wert um -0,5 Zoll)
G10 L2 P4 Y0.5 (verschiebt G57 Y-Wert um 0,5 Zoll)
G10 L2 P5 Y-0.5 (verschiebt G58 Y-Wert um -0,5 Zoll)
```

Diese haben wir in dem folgenden Programm zusammengestellt:

```
(ein Programm zum Fräsen von fünf kleinen Kreisen in Rautenform)
```

```
G10 L2 P1 X0 Y0 Z0 (sicherstellen, dass G54 Maschinen-Null ist)
G10 L2 P2 X0.5 (verschiebt den G55 X-Wert um 0,5 Zoll)
G10 L2 P3 X-0.5 (verschiebt den G56 X-Wert um -0,5 Zoll)
G10 L2 P4 Y0.5 (verschiebt G57 Y-Wert um 0,5 Zoll)
G10 L2 P5 Y-0.5 (verschiebt G58 Y-Wert um -0,5 Zoll)

G54 G0 X-0.1 Y0 Z0 (mittlerer Kreis)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G55 G0 X-0.1 Y0 Z0 (erster versetzter Kreis)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G56 G0 X-0.1 Y0 Z0 (zweiter versetzter Kreis)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G57 G0 X-0.1 Y0 Z0 (dritter versetzter Kreis)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G58 G0 X-0.1 Y0 Z0 (vierer versetzter Kreis)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G54 G0 X0 Y0 Z0

M2
```

Jetzt kommt der Zeitpunkt, an dem wir eine Reihe von G92-Offsets auf dieses Programm anwenden können. Sie werden sehen, dass es in jedem Fall auf Z0 läuft. Wenn sich die Fräse in der Nullposition befände, würde ein G92 Z1.0000 am Anfang des Programms alles um einen Zoll verschieben. Sie könnten auch das gesamte Muster in der XY-Ebene verschieben, indem Sie mit G92 einige X- und Y-Versätze hinzufügen. Wenn Sie dies tun, sollten Sie einen G92.1-Befehl kurz vor dem M2-Befehl hinzufügen, der das Programm beendet. Wenn Sie dies nicht tun, werden andere Programme, die Sie nach diesem Programm ausführen, ebenfalls diesen G92-Offset verwenden. Darüber hinaus würde es die G92-Werte zu speichern, wenn Sie die LinuxCNC herunterfahren und sie werden abgerufen, wenn Sie wieder starten.

Beispielprogramm mit G52-Offsets

(muss noch geschrieben werden)

11.2. Werkzeug Kompensation

11.2.1. Touch-Off

Mit dem Touch Off Screen in der AXIS Schnittstelle können Sie die Werkzeugtabelle automatisch aktualisieren.

Typische Schritte zum Aktualisieren der Werkzeugtabelle:

- Nach der Referenzfahrt laden Sie ein Werkzeug mit *Tn M6*, wobei *n* die Werkzeugnummer (engl. tool number) ist.
- Fahren Sie das Werkzeug mit Hilfe einer Lehre auf einen festgelegten Punkt oder machen Sie einen Testschnitt und messen Sie.
- Klicken Sie auf der Registerkarte "Manuelle Steuerung" auf den Button "Ausschalten" (oder drücken Sie die Taste "Ende" auf Ihrer Tastatur).
- Wählen Sie „Werkzeugtabelle“ im Dropdown-Feld „Koordinatensystem“ aus.
- Geben Sie das Messgerät oder die gemessene Bemaßung ein und wählen Sie OK aus.

Die Werkzeugtabelle wird mit der korrekten Z-Länge geändert, damit die DRO die richtige Z-Position anzeigt, und ein G43-Befehl wird ausgegeben, damit die neue Z-Länge des Werkzeugs in Kraft tritt. Das Antasten der Werkzeugtabelle ist nur verfügbar, wenn ein Werkzeug mit *Tn M6* geladen ist.

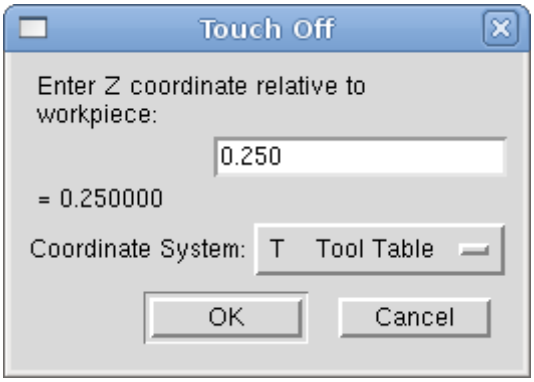


Figure 200. Touch-Off-Werkzeugtabelle

Verwendung von G10 L1/L10/L11

Die G10-Befehle L1/L10/L11 können zum Einstellen von Werkzeugtabellen-Offsets verwendet werden:

- `G10 L1 P__n__` - Setzt den/die Offset(s) auf einen Wert. Die aktuelle Position ist irrelevant (siehe G10 L1 für Einzelheiten).
- `G10 L10 P__n__` - Setzt Offset(s), so dass die aktuelle Position mit dem Gerät 1-8 ein Wert wird (siehe G10 L10 für Details).
- `G10 L11 P__n__` - Offset(s) setzen, so dass die aktuelle Position mit dem Gerät 9 ein Wert wird (siehe G10 L11 für Details).

NOTE Dies ist nur eine kurze Darstellung, genauere Erläuterungen finden Sie im Referenzhandbuch des G-Codes.

11.2.2. Werkzeugtabelle

Die "Werkzeugtabelle" ist eine Textdatei, die Informationen über jedes Werkzeug enthält. Die Datei befindet sich im gleichen Verzeichnis wie Ihre Konfiguration und heißt standardmäßig "tool.tbl". Ein Dateiname kann mit der INI-Datei [EMCIO]TOOL_TABLE festgelegt werden. Die Werkzeuge können sich in einem Werkzeugwechsler befinden oder einfach manuell geändert werden. Die Datei kann mit einem Texteditor bearbeitet werden oder mit G10 L1 aktualisiert werden. Im Abschnitt Lathe Tool Table finden Sie ein Beispiel für das Format der Drehbank-Werkzeugtabelle. Die maximale Platzanzahl beträgt 1000.

Der Tool Editor oder ein Texteditor können verwendet werden, um die Werkzeugtabelle zu bearbeiten. Wenn Sie einen Texteditor verwenden, stellen Sie sicher, dass Sie die Werkzeugtabelle in der GUI neu laden.

Werkzeugtabellen-Format

.Werkzeugtabellen-Format

T#	P#	X	Y	Z	A	B	C	U	V	W	Durchm	FA	BA	Ori	Rem
;	(keine Daten nach dem öffnenden Semikolon)														
T1	P17	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem
T2	P5	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem
T3	P12	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem

Im Allgemeinen ist das Zeilenformat der Werkzeugtabelle wie folgt:

- T - Werkzeugnummer (Werkzeugnummern müssen eindeutig sein)
- P - Taschennummer, 1-1000 (Taschennummern müssen eindeutig sein, Tasche 0 steht für die Spindel)
- X.. W - Werkzeugversatz auf vorgegebener Achse - Gleitkommazahl
- D - Werkzeugdurchmesser - Fließkommazahl, absoluter Wert
- I - Frontwinkel (nur Drehbank) - Gleitkommazahl
- J - Rückenwinkel (nur Drehmaschine) - Gleitkommazahl
- Q - Werkzeugausrichtung (nur Drehmaschine) - ganze Zahl, 0-9
- ; - Beginn des Kommentars oder der Bemerkung - Text

Werkzeugnummern sollten eindeutig sein. Zeilen, die mit einem Semikolon beginnen, werden ignoriert.

Die Einheiten für Länge, Durchmesser usw. werden in Maschineneinheiten angegeben.

Wahrscheinlich werden Sie die Werkzeugeinträge in aufsteigender Reihenfolge halten wollen, besonders wenn Sie einen zufälligen Werkzeugwechsler verwenden. Die Werkzeugtabelle erlaubt jedoch Werkzeugnummern in beliebiger Reihenfolge.

Eine Zeile kann bis zu 16 Einträge enthalten, wird aber wahrscheinlich viel weniger enthalten. Die Einträge für T (Werkzeugnummer) und P (Platznummer) sind erforderlich. Der letzte Eintrag (eine Bemerkung oder ein Kommentar, dem ein Semikolon vorangestellt ist) ist optional. Es erleichtert das Lesen, wenn die Einträge in Spalten angeordnet sind, wie in der Tabelle gezeigt, aber die einzige Formatvorschrift ist, dass nach jedem Eintrag in einer Zeile mindestens ein Leerzeichen oder Tabulator und am Ende jedes Eintrags ein Zeilenumbruch stehen muss.

Die Bedeutung der Einträge und die Art der Daten, die sie enthalten, sind wie folgt.

Werkzeugnummer (erforderlich)

Die Spalte *T* enthält die Zahl (Ganzzahl ohne Vorzeichen), die eine Codenummer für das Tool darstellt. Der Benutzer kann jeden Code für jedes Tool verwenden, solange die Codes ganze Zahlen ohne Vorzeichen sind.

Taschen-Nummer (erforderlich)

Die Spalte *P* enthält die Taschennummer (ganzzahlig ohne Vorzeichen, auch Steckplatznummer, engl. pocket number oder slot number) des Werkzeugwechslers, in dem sich das Werkzeug befindet. Die Einträge in dieser Spalte müssen alle unterschiedlich sein.

Die Platznummern beginnen in der Regel bei 1 und gehen bis zum höchsten verfügbaren Platz auf Ihrem Werkzeugwechsler. Aber nicht alle Werkzeugwechsler folgen diesem Muster. Die Platznummern richten sich nach den Nummern, die Ihr Werkzeugwechsler für die Bezeichnungen der Plätze verwendet. Das heißt also, dass die von Ihnen verwendeten Platznummern durch das Nummerierungsschema Ihres Werkzeugwechslers bestimmt werden, und dass die von Ihnen verwendeten Platznummern auf Ihrer Maschine Sinn machen müssen.

Daten-Offset-Nummern (optional)

Die Spalten *Datenversatz* (XYZABCUVW) enthalten reelle Zahlen, die Werkzeugversätze in jeder Achse darstellen. Diese Zahl wird verwendet, wenn die Werkzeuglängenkorrekturen verwendet werden und dieses Werkzeug ausgewählt ist. Diese Zahlen können positiv, null oder negativ sein, und sind eigentlich völlig optional. Allerdings sollten Sie hier mindestens einen Eintrag vornehmen, da es sonst wenig Sinn macht, einen Eintrag in der Werkzeugtabelle vorzunehmen.

Bei einer typischen Fräsmaschine benötigen Sie wahrscheinlich einen Eintrag für Z (Werkzeuglängenkorrektur). Bei einer typischen Drehmaschine benötigen Sie wahrscheinlich einen Eintrag für X (X-Werkzeugkorrektur) und Z (Z-Werkzeugkorrektur). Bei einer typischen Fräsmaschine, die eine Fräserdurchmesserkompensation verwendet, möchten Sie wahrscheinlich auch einen Eintrag für D (Fräserdurchmesser) hinzufügen. In einer typischen Drehmaschine, die eine Werkzeugdurchmesserkompensation (tool comp) verwendet, möchten Sie wahrscheinlich auch einen Eintrag für D (Werkzeugdurchmesser) hinzufügen.

Eine Drehmaschine erfordert auch einige zusätzliche Informationen, um die Form und Ausrichtung des Werkzeugs zu beschreiben. Daher möchten Sie wahrscheinlich Einträge für I (vorderer Werkzeugwinkel) und J (hinterer Werkzeugwinkel) haben. Wahrscheinlich möchten Sie auch einen Eintrag für Q (Werkzeugausrichtung).

Siehe das Kapitel Informationen für Nutzer von Drehmaschinen für weitere Details.

Die Spalte "Durchmesser" enthält eine reelle Zahl. Diese Zahl wird nur verwendet, wenn die

Fräskompensation für dieses Werkzeug aktiviert ist. Wenn die programmierte Bahn während der Kompensation die Kante des zu schneidenden Materials ist, sollte dies eine positive reelle Zahl sein, die den gemessenen Durchmesser des Werkzeugs darstellt. Wenn die programmierte Bahn während der Kompensation die Bahn eines Werkzeugs ist, dessen Durchmesser nominal ist, sollte dies eine kleine Zahl sein (positiv oder negativ, aber nahe Null), die nur den Unterschied zwischen dem gemessenen Durchmesser des Werkzeugs und dem nominalen Durchmesser darstellt. Wenn die Fräser-Korrektur nicht mit einem Werkzeug verwendet wird, spielt es keine Rolle, welche Zahl in dieser Spalte steht.

Die Spalte "Kommentar" kann optional zur Beschreibung des Werkzeugs verwendet werden. Jede Art von Beschreibung ist zulässig. Diese Spalte ist nur für den menschlichen Leser gedacht. Dem Kommentar muss ein Semikolon vorangestellt werden.

NOTE Frühere Versionen von LinuxCNC hatte zwei verschiedene Werkzeug-Tabelle Formate für Fräsen und Drehen, aber seit der 2.4.x Release, wird dasselbe Werkzeug-Tabellen-Format für alle Maschinen verwendet.

Werkzeug IO (engl. Tool IO)

Das nicht-Echtzeit-Programm **iocontrol** wird üblicherweise für die Werkzeugwechsler-Verwaltung (und andere E/A (I/O)-Funktionen zum Aktivieren von LinuxCNC und die Steuerung von Kühlmittel-Hardware) verwendet. Die HAL-Pins, die für die Werkzeugverwaltung verwendet werden, sind mit dem Präfix **iocontrol.0.** versehen.

Ein G-Code **T**-Befehl setzt den HAL-Ausgangspin `iocontrol.0.tool-prepare`. Der HAL-Eingangspin `iocontrol.0.tool-prepared` muss durch externe HAL-Logik gesetzt werden, um die Werkzeugvorbereitung abzuschließen, was zu einem anschließenden Reset des Tool-Prepare-Pins führt.

Ein G-Code **M6**-Befehl aktiviert den HAL-Ausgangspin `iocontrol.0.tool-change`. Der zugehörige HAL-Eingangs-Pin, `iocontrol.0.tool-prepared`, muss durch externe HAL-Logik gesetzt werden, um den Abschluss des Werkzeugwechsels anzuzeigen, was zu einem anschließenden Reset des Tool-Change-Pins führt.

Der Zugriff auf die Werkzeugdaten erfolgt über einen geordneten Index (idx), der vom Typ des Werkzeugwechslers abhängt, der durch `[EMCIO]RANDOM_TOOLCHANGER=`*type* festgelegt ist.

1. Bei `RANDOM_TOOLCHANGER = 0` (0 ist die Standardeinstellung und gibt einen nicht zufälligen Werkzeugwechsler an) ist idx eine Zahl, zur Angabe der Reihenfolge, in der die Werkzeugdaten geladen wurden.
2. Bei `RANDOM_TOOLCHANGER = 1` ist idx die **aktuelle** Platznummer für die Werkzeugnummer, die durch den G-Code-Befehl zur Werkzeugauswahl **T*n*** festgelegt wurde.

Das io-Programm bietet HAL Ausgangsstifte, um die Verwaltung des Werkzeugwechslers zu erleichtern:

1. **iocontrol.0.tool-prep-number**
2. **iocontrol.0.tool-prep-index**
3. **iocontrol.0.tool-prep-pocket**
4. **iocontrol.0.tool-from-pocket**

E/A (I/O) für nicht-zufälligen Werkzeugwechsler

1. Werkzeugnummer *n*==0 zeigt an, dass kein Werkzeug vorhanden ist.
2. Die Platznummer für ein Werkzeug wird beim Laden/Nachladen der Werkzeugdaten aus der Datenquelle ([EMCIO]TOOL_TABLE oder [EMCIO]DB_PROGRAM) festgelegt.
3. Beim Befehl G-Code **T*n*** (*n* != 0):
   a. **iocontrol.0.tool-prep-index** = *idx* (Index basierend auf der Tooldaten-Ladesefolge)
   b. **iocontrol.0.tool-prep-number** = *n*
   c. **iocontrol.0.tool-prep-pocket** = die feste Platz-/Taschen-Nummer für *n*
4. Beim Befehl G-code **T0** (*n* == 0 entfernen):
   a. **iocontrol.0.tool-prep-index** = 0
   b. **iocontrol.0.tool-prep-number** = 0
   c. **iocontrol.0.tool-prep-pocket** = 0
5. Bei M-Code **M6** (nach iocontrol.0.tool-changed pin 0-→1):
   a. **iocontrol.0.tool-from-pocket** = Nummer der Tasche, die zum Abrufen des Werkzeugs verwendet wird

E/A (I/O) für zufälligen Werkzeugwechsler

1. Die Werkzeugnummer *n*==0 ist **nicht speziell**.
2. Die Taschennummer 0 ist **speziell**, da sie die **Spindel** anzeigt.
3. Die **aktuelle** Platznummer für Werkzeug *n* ist der Werkzeugdatenindex (*idx*) für Werkzeug *n*.
4. Bei G-Code Befehl **T*n***:
   a. **iocontrol.0.tool-prep-index** = Tooldatenindex (idx) für Werkzeug *n*
   b. **iocontrol.0.tool-prep-number** = *n*
   c. **iocontrol.0.tool-prep-pocket** = Platznummer für Werkzeug *n*
5. Bei M-Code **M6** (nach iocontrol.0.tool-changed pin 0-→1):
   a. **iocontrol.0.tool-from-pocket** = Nummer der Tasche, die zum Abrufen des Werkzeugs verwendet wird

NOTE Beim Start ist **iocontrol.0.tool-from-pocket** = 0. Ein M61Q*n* (*n*!=0) Befehl ändert **iocontrol.0.tool-from-pocket** nicht. Ein M61Q0 (n==0) Befehl setzt **iocontrol.0.tool-from-pocket** auf 0.

Werkzeugwechsler

LinuxCNC unterstützt drei Arten von Werkzeugwechslern: *manuell*, *zufällige Position* und *nicht zufällige oder feste Position*. Informationen über die Konfiguration eines LinuxCNC Werkzeugwechsler ist in dem Abschnitt EMCIO des INI-Kapitels.

Manueller Werkzeugwechsler

Ein manueller Werkzeugwechsler (Sie wechseln das Werkzeug von Hand) wird wie ein Festplatz-Werkzeugwechsler behandelt. Manuelle Werkzeugwechsel können durch eine HAL-Konfiguration unterstützt werden, die das nicht-Echtzeit-Programm **hal_manualtoolchange** verwendet und normalerweise in einer INI-Datei mit INI-Anweisungen angegeben wird:

```
[HAL]
HALFILE = axis_manualtoolchange.hal
```

Werkzeugwechsler mit festen Plätzen

Werkzeugwechsler mit fester Positionierung bringen die Werkzeuge immer in eine feste Position im Werkzeugwechsler zurück. Dies würde auch Designs wie Drehmaschine Revolver umfassen. Wenn LinuxCNC für eine feste Position Werkzeugwechsler konfiguriert ist die *P*-Nummer nicht intern verwendet wird (aber gelesen, erhalten und neu geschrieben) von LinuxCNC, so können Sie P für jede Buchhaltung Nummer, die Sie wollen.

NOTE

Bei Verwendung von `[EMCIO]RANDOM_TOOLCHANGER = 0` (Standardeinstellung) ist die *P* -Taschennummer ein Parameter der Werkzeugdaten (engl. tooldata), die aus der Werkzeugdatenquelle (engl. tooldata source) (`[EMCIO]TOOL_TABLE` oder `[EMCIO]DB_PROGRAM`) abgerufen werden. In vielen Anwendungen ist es fest, aber es kann durch Bearbeitungen der `[EMCIO]TOOL_TABLE` oder programmatisch geändert werden, wenn das `[EMCIO]DB_PROGRAM` verwendet wird. LinuxCNC schiebt Updates auf die Datenquelle (`[EMCIO]TOOL_TABLE` oder `[EMCIO]DB_PROGRAM`) für G-Codes G10L1, G10L10, G10L11, M61. LinuxCNC kann Updates zu Werkzeugdaten aus der Datenquelle beziehen durch UI (User-Interface) Befehle (Python Beispiel: `linuxcnc.command().load_tool_table()`) oder durch G-Code: `G10L0`.

Werkzeugwechsler mit zufälliger Position

Zufallswerkzeugwechsler (`[EMCIO]RANDOM_TOOLCHANGER = 1`) tauschen das Werkzeug in der Spindel mit dem Werkzeug im Wechsler aus. Mit dieser Art von Werkzeugwechsler wird das Werkzeug immer in einer anderen Tasche nach einem Werkzeugwechsel sein. Wenn ein Werkzeug gewechselt wird, schreibt LinuxCNC die Platznummer neu, um den Überblick zu behalten, wo die Werkzeuge sind. T kann eine beliebige Zahl sein, aber P muss eine Zahl sein, die für die Maschine Sinn Macht.

11.2.3. Werkzeuglängenkompensation

Die Werkzeuglängenkorrekturen werden als positive Zahlen in der Werkzeugtabelle angegeben. Eine Werkzeugkorrektur wird mit G43 H*n* programmiert, wobei *n* die Indexnummer des gewünschten Werkzeugs in der Werkzeugtabelle ist. Es ist vorgesehen, dass alle Einträge in der Werkzeugtabelle positiv sind. Der Wert von H wird geprüft, er muss beim Lesen eine nicht-negative ganze Zahl sein. Der Interpreter verhält sich wie folgt:

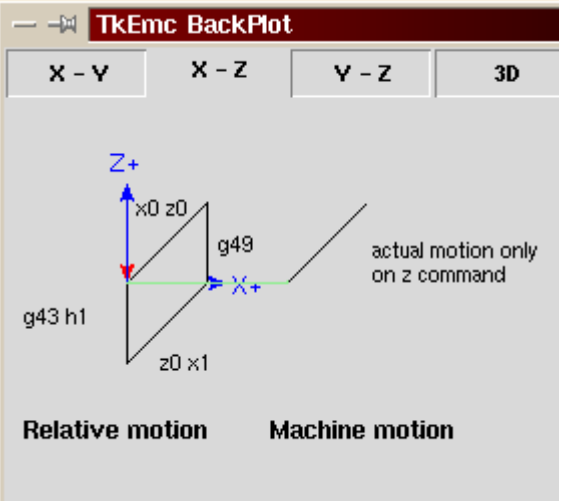
1. Wenn G43 H*n* programmiert ist, erfolgt ein Aufruf der Funktion `USE_TOOL_LENGTH_OFFSET(`__length__)` (wobei *length* die aus der Werkzeugtabelle gelesene Längendifferenz des indizierten Werkzeugs *n* ist), tool_length_offset wird im Maschineneinstellungsmodell neu positioniert und der Wert von current_z im Modell wird

angepasst. Beachten Sie, dass *n* nicht mit der Steckplatz-Nummer des Werkzeugs übereinstimmen muss, das sich gerade in der Spindel befindet.

2. Wenn G49 programmiert ist, wird `USE_TOOL_LENGTH_OFFSET(0.0)` aufgerufen, tool_length_offset wird in der Maschineneinstellungsvorlage auf 0.0 zurückgesetzt und der aktuelle Wert von current_z im Modell wird angepasst. Die Auswirkung der Werkzeuglängenkompensation ist in der folgenden Abbildung zu sehen. Beachten Sie, dass die Werkzeuglänge von Z abgezogen wird, so dass der programmierte Kontrollpunkt mit der Spitze des Werkzeugs übereinstimmt. Beachten Sie auch, dass die Auswirkung der Längenkompensation sofort sichtbar ist, wenn Sie die Position von Z als relative Koordinate sehen, aber sie hat keine Auswirkung auf die tatsächliche Maschinenposition, bis eine Z-Bewegung programmiert wird.

Testprogramm für die Werkzeuglänge. Werkzeug #1 ist einen Zoll lang.

```
N01 G1 F15 X0 Y0 Z0
N02 G43 H1 Z0 X1
N03 G49 X0 Z0
N04 G0 X2
N05 G1 G43 H1 G4 P10 Z0 X3
N06 G49 X2 Z0
N07 G0 X0
```



Mit diesem Programm wird die Maschine in den meisten Fällen den Offset in Form einer Rampe während der Bewegung in xyz nach dem Wort G43 anwenden.

11.2.4. Fräserradius-Kompensation

Die Fräserkompensation ermöglicht es dem Programmierer, den Werkzeugweg zu programmieren, ohne den genauen Werkzeugdurchmesser zu kennen. Der einzige Vorbehalt ist, dass der Programmierer den Lead in Move so programmieren muss, dass er mindestens so lang ist wie der größte Werkzeugradius, der verwendet werden könnte.

Es gibt zwei mögliche Pfade, die der Fräser einschlagen kann, da der Fräserausgleich auf der linken oder der rechten Seite einer Linie vorgenommen werden kann, wenn man die Bewegungsrichtung des Fräsers von hinten betrachtet. Um dies zu veranschaulichen, stellen Sie sich vor, Sie stünden auf dem Werkstück und gingen hinter dem Werkzeug, während es sich über das Werkstück bewegt. G41 ist Ihre

linke Seite der Linie und G42 ist die rechte Seite der Linie.

Der Endpunkt jeder Bewegung hängt von der nächsten Bewegung ab. Wenn die nächste Bewegung eine Außenecke erzeugt, erfolgt die Bewegung zum Endpunkt der kompensierten Schnittlinie. Wenn die nächste Bewegung eine Innenecke erzeugt, wird die Bewegung kurz angehalten, um das Teil nicht auszuschneiden. Die folgende Abbildung zeigt, wie die kompensierte Bewegung je nach der nächsten Bewegung an verschiedenen Punkten stoppt.

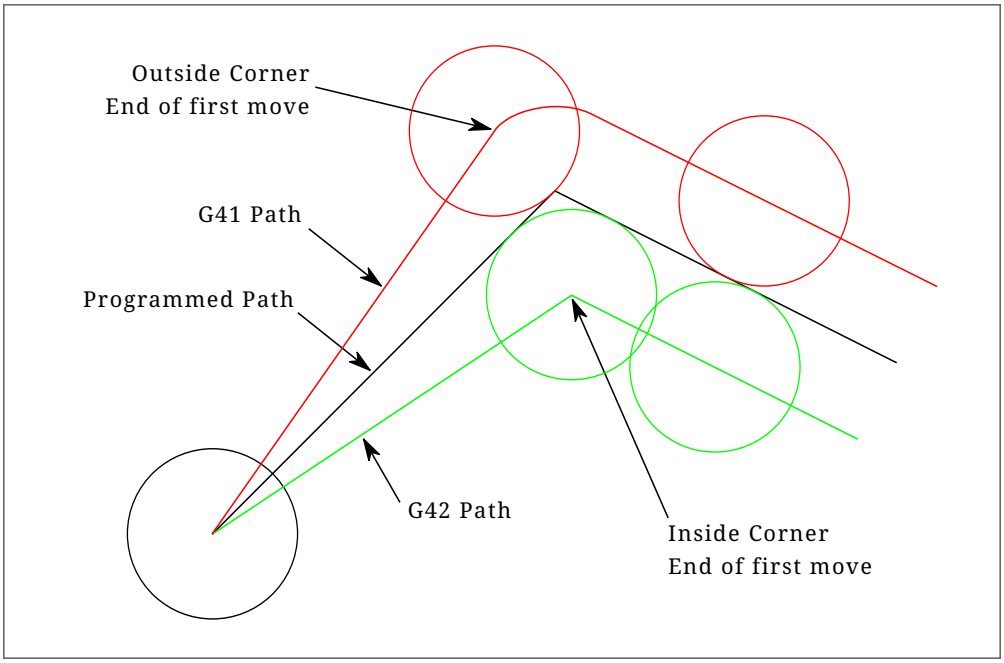


Figure 201. Ausgleich am Endpunkt (engl. Compensation End Point)

Übersicht

Werkzeugtabelle

Die Fräserkompensation verwendet die Daten aus der Werkzeugtabelle, um den benötigten Versatz zu bestimmen. Die Daten können zur Laufzeit mit G10 L1 eingestellt werden.

Einstiegsbewegungen programmieren

Jede Bewegung, die lang genug ist, um die Kompensation durchzuführen, kann als Eingangsbewegung verwendet werden. Die Mindestlänge ist der Radius des Fräsers. Dies kann eine Eilgangbewegung über dem Werkstück sein. Wenn mehrere Eilgänge nach einem G41/42 ausgeführt werden, fährt nur der letzte das Werkzeug in die kompensierte Position.

In der folgenden Abbildung sehen Sie, dass die Einfahrbewegung rechts von der Linie kompensiert wird. Dadurch befindet sich der Mittelpunkt des Werkzeugs in diesem Fall rechts von X0. Wenn Sie ein Profil programmieren würden und das Ende bei X0 liegt, würde das resultierende Profil aufgrund des Versatzes der Einfahrbewegung eine Beule hinterlassen.

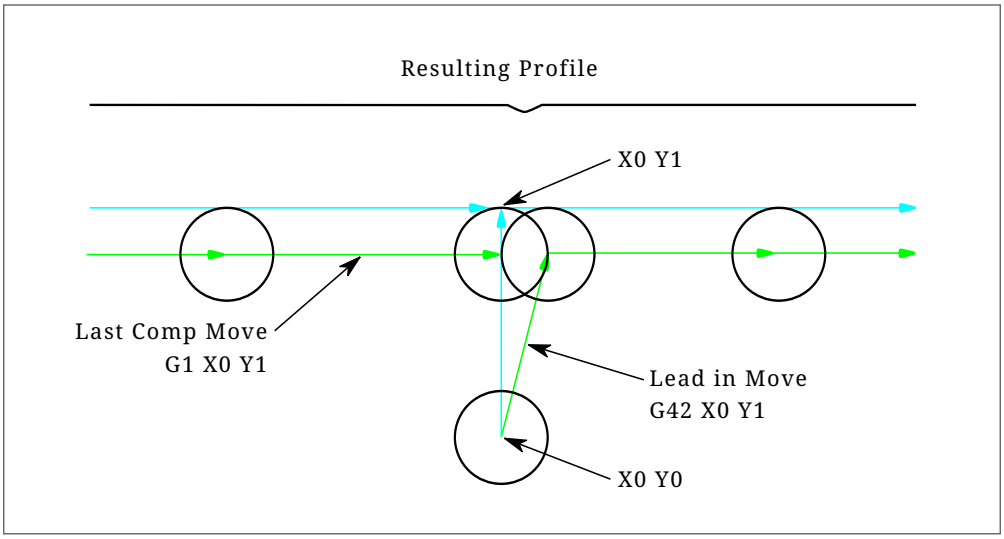


Figure 202. Eingangs-Bewegung (engl. entry move)

Z-Bewegung

Die Bewegung der Z-Achse kann erfolgen, während die Kontur in der XY-Ebene verfolgt wird. Teile der Kontur können übersprungen werden, indem die Z-Achse über dem Teil zurückgezogen und am nächsten Startpunkt wieder ausgefahren wird.

Eilgänge

Eilgänge können programmiert werden, während die Kompensation eingeschaltet ist.

Gute Praktiken

Starten Sie ein Programm mit G40, um sicherzustellen, dass die Kompensation ausgeschaltet ist.

Beispiele

Beispiel für ein äußeres Profil

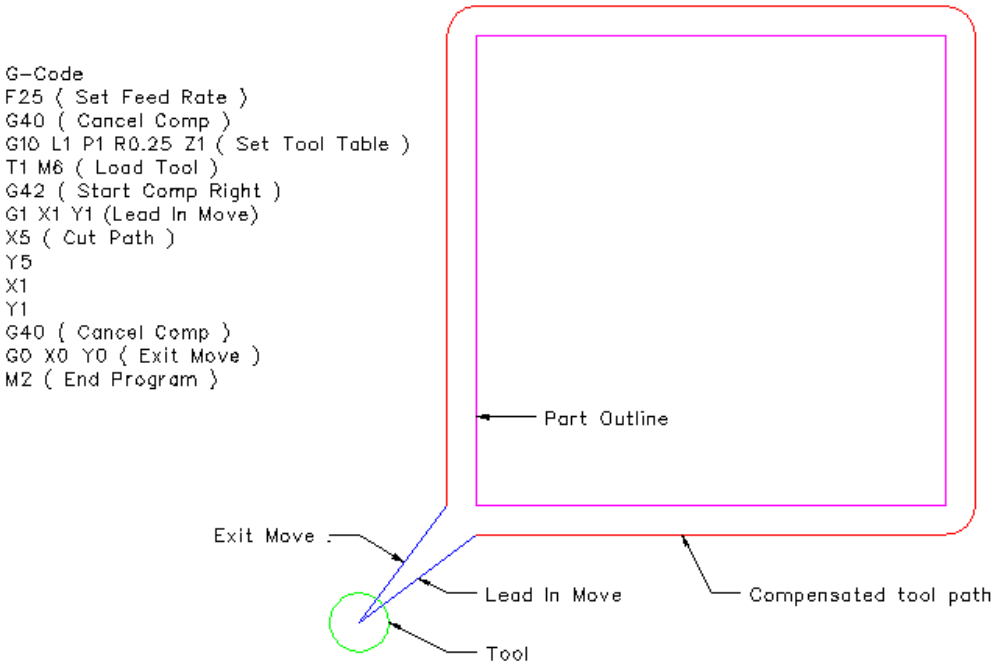


Figure 203. Äußeres Profil

Beispiel für ein Innenprofil

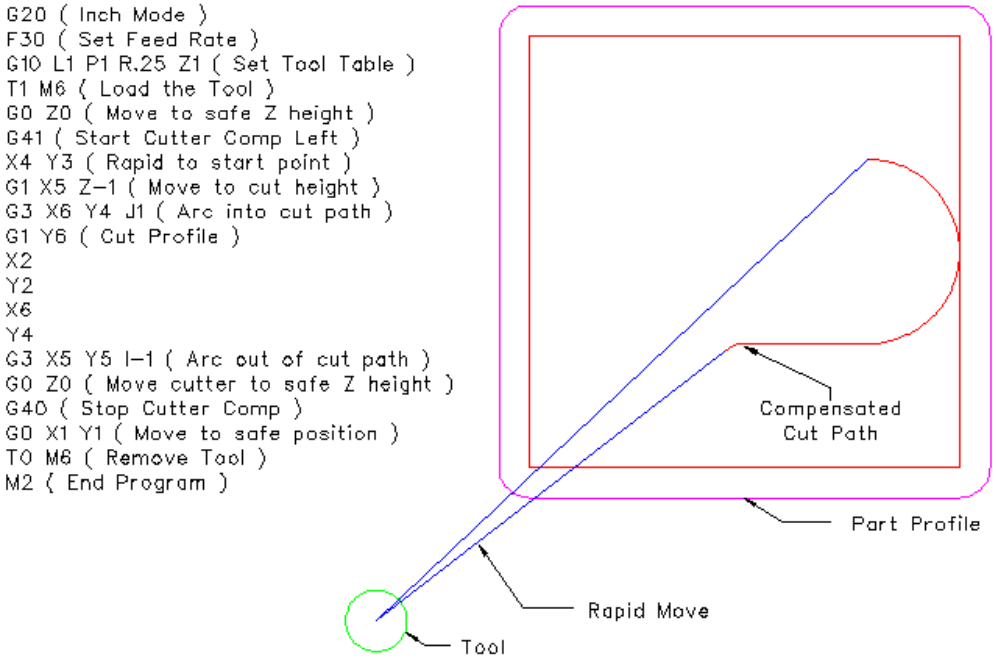


Figure 204. Innenprofil

11.3. GUI zur Werkzeug-Bearbeitung

11.3.1. Übersicht

NOTE Die hier beschriebenen Tooledit-Elemente sind seit Version 2.5.1 verfügbar. In Version 2.5.0 erlaubt die grafische Benutzeroberfläche diese Anpassungen nicht.

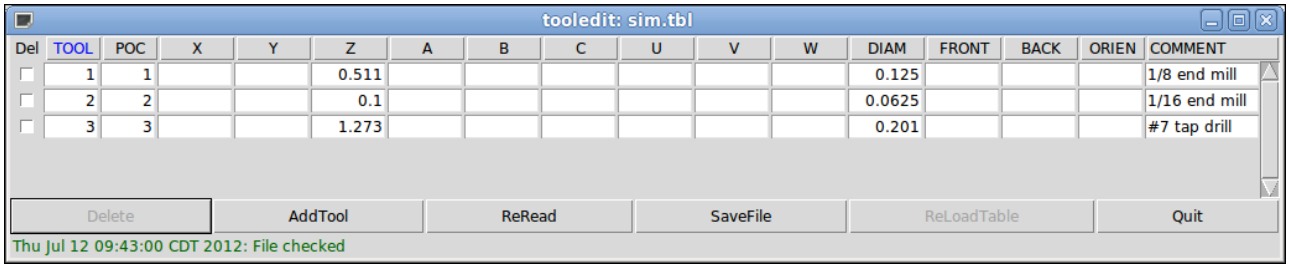


Figure 205. Tool Edit GUI - Überblick

Das Programm *tooledit* kann die Werkzeugtabellendatei mit bearbeiteten Änderungen aktualisieren, indem es die Schaltfläche SaveFile verwendet. Die Schaltfläche SaveFile aktualisiert die Systemdatei, aber eine separate Aktion ist erforderlich, um die Werkzeugtabelle Daten von einem laufenden LinuxCNC Instanz verwendet aktualisieren. Mit der AXIS GUI können sowohl die Datei als auch die aktuellen, von LinuxCNC verwendeten Werkzeugtabellendaten mit der Schaltfläche ReloadTable aktualisiert werden. Diese Schaltfläche ist nur aktiviert, wenn die Maschine eingeschaltet und im Leerlauf ist.

11.3.2. Spaltensortierung

Die Anzeige der Werkzeugtabelle kann nach jeder Spalte in aufsteigender Reihenfolge sortiert werden, indem Sie auf die Spaltenüberschrift klicken. Ein zweiter Klick sortiert in absteigender Reihenfolge. Die Spaltensortierung erfordert, dass die Maschine mit der Standard-Tcl-Version >= 8.5 konfiguriert ist.

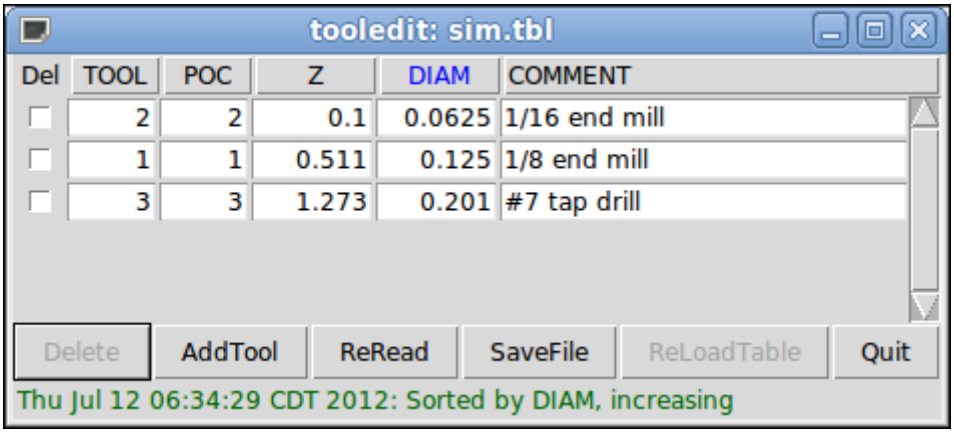


Figure 206. Tool Edit GUI - Spaltensortierung

In Ubuntu Lucid 10.04 ist Tcl/Tk8.4 standardmäßig installiert. Die Installation wird wie folgt durchgeführt:

```
sudo apt-get install tcl8.5 tk8.5
```

Je nachdem, welche anderen Anwendungen auf dem System installiert sind, kann es notwendig sein, Tcl/Tk8.5 mit den Befehlen zu aktivieren:

```
sudo update-alternatives --config tclsh  ;# select the option for tclsh8.5
sudo update-alternatives --config wish   ;# select the option for wish8.5
```

11.3.3. Spaltenauswahl

Standardmäßig zeigt das Programm *tooledit* alle möglichen Spalten der Werkzeugtabelle an. Da nur wenige Maschinen alle Parameter verwenden, können die angezeigten Spalten mit der folgenden INI-Datei-Einstellung eingeschränkt werden:

Syntax der INI-Datei

```
[DISPLAY]
TOOL_EDITOR = tooledit column_name column_name ...
```

Beispiel für Z- und DIAM-Spalten

```
[DISPLAY]
TOOL_EDITOR = tooledit Z DIAM
```

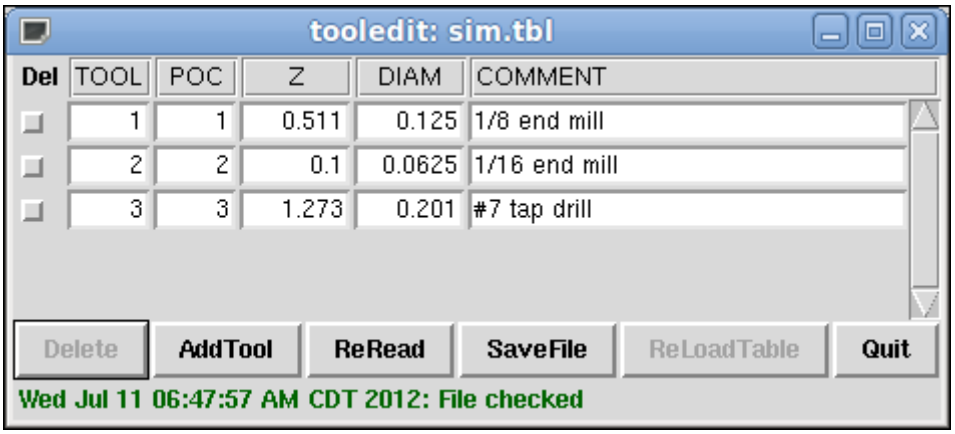


Figure 207. Tool Edit GUI - Spaltenauswahl Beispiel

11.3.4. Eigenständige Verwendung

Das Programm "tooledit" kann auch als eigenständiges Programm aufgerufen werden. Wenn sich das Programm beispielsweise im Benutzerpfad befindet, zeigt die Eingabe von "tooledit" die Verwendungssyntax an:

Eigenständig (engl. stand alone)

```
tooledit
Usage:
       tooledit filename
       tooledit [column_1 ... column_n] filename

Gültige Spaltennamen sind: x y z a b c u v w diam front back orien
```

Um eine eigenständige *tooledit* mit einem laufenden LinuxCNC-Anwendung zu synchronisieren, muss der Dateiname auf die gleiche [EMCIO]TOOL_TABLE Dateiname in der LinuxCNC INI-Datei angegeben aufzulösen.

Wenn Sie das Programm *tooledit* verwenden, während LinuxCNC läuft, können die Ausführung von G-Code-Befehlen oder andere Programme die Werkzeugtabellendaten und die Werkzeugtabellendatei verändern. Dateiänderungen werden von *tooledit* erkannt und eine Meldung wird angezeigt:

```
Warnung: Datei von einem anderen Prozess geändert (engl: File changed by another process)
```

Die Anzeige der Werkzeugtabelle *tooledit* kann mit dem ReRead Button aktualisiert werden, um die geänderte Datei zu lesen.

Die Werkzeugtabelle wird in der INI-Datei mit einem Eintrag angegeben:

```
[EMCIO]TOOL_TABLE = tool_table_filename
```

Die Werkzeugtabellendatei kann mit jedem einfachen Texteditor (nicht mit einem Textverarbeitungsprogramm) bearbeitet werden.

Das AXIS GUI kann optional eine INI-Datei-Einstellung verwenden, um das Werkzeug-Editor-Programm festzulegen:

```
[DISPLAY]TOOL_EDITOR = path_to_editor_program
```

Standardmäßig wird das Programm mit dem Namen "tooledit" verwendet. Dieser Editor unterstützt alle Parameter der Werkzeugtabelle, ermöglicht das Hinzufügen und Löschen von Werkzeugeinträgen und führt eine Reihe von Gültigkeitsprüfungen der Parameterwerte durch.

11.4. Überblick zur G-Code Programmierung

11.4.1. Übersicht

Die LinuxCNC G-Code Sprache basiert auf der RS274/NGC Sprache. Die G-Code-Sprache basiert auf Codezeilen. Jede Zeile (auch *Block* genannt) kann Befehle enthalten, um mehrere verschiedene Dinge zu tun. Codezeilen können in einer Datei gesammelt werden, um ein Programm zu erstellen.

Eine typische Codezeile besteht aus einer optionalen Zeilennummer am Anfang, gefolgt von einem oder mehreren *Wörtern*. Ein Wort besteht aus einem Buchstaben gefolgt von einer Zahl (oder etwas, das als Zahl ausgewertet werden kann). Ein Wort kann entweder einen Befehl geben oder ein Argument für einen Befehl darstellen. Zum Beispiel ist *G1 X3* eine gültige Codezeile mit zwei Wörtern. *G1* ist ein Befehl, der bedeutet *fahre in einer geraden Linie mit der programmierten Vorschubgeschwindigkeit zum programmierten Endpunkt*, und *X3* liefert einen Argumentwert (der Wert von X sollte am Ende der Bewegung 3 sein). Die meisten LinuxCNC G-Code Befehle beginnen entweder mit G oder M (für General und Miscellaneous). Die Wörter für diese Befehle werden *G-Codes* und *M-Codes* genannt. Häufig sind auch Unterprogrammcodes, die mit *o-* beginnen, die als *o-Codes* bezeichnet werden.

Die LinuxCNC Sprache hat keinen Indikator für den Start eines Programms. Der Interpreter arbeitet jedoch mit Dateien. Ein einzelnes Programm kann in einer einzigen Datei stehen, oder ein Programm kann über mehrere Dateien verteilt sein. Eine Datei kann auf folgende Weise mit Prozent-Zeichen abgegrenzt werden. Die erste nicht leere Zeile einer Datei kann nichts anderes als ein Prozentzeichen, %,

enthalten, möglicherweise umgeben von Leerzeichen, und später in der Datei (normalerweise am Ende der Datei) kann es eine ähnliche Zeile geben. Die Abgrenzung einer Datei mit Prozentzeichen ist optional, wenn die Datei ein *M2* oder *M30* enthält, ist aber erforderlich, wenn nicht. Ein Fehler wird gemeldet, wenn eine Datei am Anfang, aber nicht am Ende eine Prozentzeile enthält. Der nützliche Inhalt einer Datei, die durch Prozentzeichen abgegrenzt ist, hört nach der zweiten Prozentzeile auf. Alles, was danach kommt, wird ignoriert.

Die LinuxCNC G-Code Sprache hat zwei Befehle (*M2* oder *M30*), von denen jeder ein Programm beendet. Ein Programm kann vor dem Ende einer Datei enden. Zeilen einer Datei, die nach dem Ende eines Programms stehen, werden nicht ausgeführt. Der Interpreter liest sie nicht einmal.

11.4.2. Format einer Zeile

Eine zulässige Eingabezeile besteht der Reihe nach aus den folgenden Zeichen, wobei die Anzahl der in einer Zeile zulässigen Zeichen begrenzt ist (derzeit 256).

1. ein optionales Blocklöschzeichen, das ein Schrägstrich / ist.
2. eine optionale Zeilennummer.
3. Irgendeine Anzahl an:
    1. Wörtern,
    2. Parameter-Einstellungen,
    3. Unterprogramm Codes, und
    4. Kommentaren.
4. eine Zeilenende-Markierung (Wagenrücklauf oder Zeilenvorschub oder beides).

Jede nicht ausdrücklich erlaubte Eingabe ist illegal und führt zu einer Fehlermeldung des Interpreters.

Leerzeichen und Tabulatoren sind an jeder Stelle einer Codezeile erlaubt und ändern die Bedeutung der Zeile nicht, außer innerhalb von Kommentaren. Dies macht einige seltsam aussehende Eingaben legal. Die Zeile *G0X +0. 12 34Y 7* ist zum Beispiel äquivalent zu *G0 x+0.1234 Y7*.

Leerzeilen sind in der Eingabe erlaubt. Sie sind zu ignorieren.

Bei der Eingabe wird nicht zwischen Groß- und Kleinschreibung unterschieden, außer in Kommentaren, d. h. jeder Buchstabe außerhalb eines Kommentars kann groß- oder kleingeschrieben sein, ohne dass sich die Bedeutung einer Zeile ändert.

/: Block löschen

Das optionale Zeichen zum Löschen von Blöcken, der Schrägstrich /, kann, wenn er an erster Stelle in einer Zeile steht, von einigen Benutzeroberflächen verwendet werden, um Codezeilen bei Bedarf zu überspringen. In Axis schaltet die Tastenkombination Alt-m-/ die Blocklöschung ein und aus. Wenn die Blocklöschung aktiviert ist, werden alle Zeilen, die mit dem Schrägstrich / beginnen, übersprungen.

In AXIS ist es auch möglich, das Löschen von Blöcken mit dem folgenden Symbol zu aktivieren:

AXIS-Block Löschsymbol



Optionale Zeilennummer

Eine Zeilennummer ist der Buchstabe N, gefolgt von einer ganzen Zahl ohne Vorzeichen, optional gefolgt von einem Punkt und einer weiteren ganzen Zahl ohne Vorzeichen. Zum Beispiel sind *N1234* und *N56.78* gültige Zeilennummern. Sie können wiederholt oder außer der Reihe verwendet werden, obwohl dies in der Regel vermieden werden sollte. Zeilennummern können auch übersprungen werden, was ebenfalls gängige Praxis ist. Eine Zeilennummer muss nicht zwingend verwendet werden, aber sie muss an der richtigen Stelle stehen, wenn sie verwendet wird.

NOTE Zeilennummern werden nicht empfohlen. Siehe Best Practices.

Wörter, Parameter, Unterroutinen, Kommentare

Wörter

Ein Wort ist ein Buchstabe außer N oder O ("o"), gefolgt von einer Gleitkommazahl.

Wörter können mit jedem der in der folgenden Tabelle aufgeführten Buchstaben beginnen. Die Tabelle enthält der Vollständigkeit halber auch N und O, obwohl Zeilennummern und Programmfluss Parameter, wie oben definiert, keine Wörter sind. Mehrere Buchstaben (I, J, K, L, P, R) können in verschiedenen Zusammenhängen unterschiedliche Bedeutungen haben. Buchstaben, die sich auf Achsennamen beziehen, gelten nicht für eine Maschine, die nicht über die entsprechende Achse verfügt.

Table 81. Wörter und ihre Bedeutungen

Buchstabe	Bedeutung
A	A-Achse der Maschine
B	B-Achse der Maschine
C	C-Achse der Maschine
D	Tool radius compensation number.
	Start angle offset from the spindle index pulse, in degrees, for G33 and G76 threading.
F	Vorschubgeschwindigkeit
G	Allgemeine Funktion (siehe Tabelle G-Code Modale Gruppen)
H	Werkzeuglängen-Offset-Index
I	X-Versatz für Bögen und G87-Festzyklen
J	Y-Versatz für Bögen und G87-Festzyklen

Buchstabe	Bedeutung
K	Z-Versatz für Bögen und G87-Konservenzyklen.
	Spindel-Bewegungs-Verhältnis für G33 synchronisierte Bewegungen.
L	generisches Parameterwort für G10, M66 und andere
M	Verschiedene Funktionen (siehe Tabelle M-Code Modale Gruppen)
N	Zeilennummer (nicht empfohlen, siehe Best Practices)
O	o-Codes für die Programmflusssteuerung (siehe o-Codes)
P	Verweilzeit in Festzyklen und mit G4.
	Schlüssel wird mit G10 verwendet.
Q	Vorschubinkrement in G73, G83 Festzyklen
R	Bogenradius oder Festzyklusebene
S	Spindeldrehzahl
T	Werkzeugauswahl
U	U-Achse der Maschine
V	V-Achse der Maschine
W	W-Achse der Maschine
X	X-Achse der Maschine
Y	Y-Achse der Maschine
Z	Z-Achse der Maschine

Parameter

Parameter werden mit einem "#"-Symbol vor ihnen identifiziert. Siehe den Abschnitt zu Parametern.

Unterprogramm Codes

Auch als "o-Codes" bezeichnet, werden diese Programm-Kontrollfluss Kommandos (wie z.B. if-else logic und aufrufbare Unterprogramme (engl. callable subroutines)) angeboten und auf der Seite zu o-Codes und auch unten im Abschnitt Subroutine-Codes und Parameter beschrieben.

NOTE o-Codes werden manchmal auch o-Worte genannt.

Kommentare

Kommentare können in eine Zeile mit Klammern () oder für den Rest einer Zeile mit einem Semikolon

eingebettet werden. Es gibt auch "aktive" Kommentare wie MSG, DEBUG, usw. Siehe die Abschnitt zu den Bemerkungen.

Markierung des Zeilenendes

Dies ist eine beliebige Kombination der Zeichen für Wagenrücklauf (engl. carriage return) oder Zeilenvorschub (engl. line feed).

11.4.3. Zahlen

Die folgenden Regeln werden für (explizite) Zahlen verwendet. In diesen Regeln ist eine Ziffer ein einzelnes Zeichen zwischen 0 und 9.

- Eine Nummer besteht aus:
  - ein optionales Plus- oder Minuszeichen, gefolgt von
  - Null bis viele Ziffern, eventuell gefolgt von
  - eine Dezimalstelle, gefolgt von
  - Null bis viele Ziffern - vorausgesetzt, die Zahl enthält mindestens eine Ziffer.
- Es gibt zwei Arten von Zahlen:
  - Ganze Zahlen, die keinen Dezimalpunkt haben,
  - Dezimalzahlen, die einen Dezimalpunkt haben.
- Zahlen können eine beliebige Anzahl von Ziffern haben, vorbehaltlich der Begrenzung der Zeilenlänge. Es werden jedoch nur etwa siebzehn signifikante Stellen beibehalten (ausreichend für alle bekannten Anwendungen).
- Eine Zahl ungleich Null ohne Vorzeichen, aber das erste Zeichen wird als positiv angenommen.

Beachten Sie, dass Anfangs- (vor dem Dezimalpunkt und der ersten Nicht-Null-Stelle) und Nachnullen (nach dem Dezimalpunkt und der letzten Nicht-Null-Stelle) erlaubt, aber nicht erforderlich sind. Eine Zahl, die mit Anfangs- oder Nachnullen geschrieben wird, hat beim Lesen denselben Wert, als ob die zusätzlichen Nullen nicht vorhanden wären.

Zahlen, die für bestimmte Zwecke in RS274/NGC verwendet werden, sind oft auf eine endliche Menge von Werten oder auf einen Wertebereich beschränkt. Bei vielen Verwendungszwecken müssen Dezimalzahlen nahe an Ganzzahlen liegen; dazu gehören die Werte von Indizes (z. B. für Parameter und Karussellplatznummern), M-Codes und G-Codes multipliziert mit zehn. Eine Dezimalzahl, die eine ganze Zahl darstellen soll, gilt als nahe genug, wenn sie innerhalb von 0,0001 eines ganzzahligen Wertes liegt.

11.4.4. Parameter

Die Sprache RS274/NGC unterstützt *Parameter* - was in anderen Programmiersprachen als *Variablen* bezeichnet würde. Es gibt mehrere Arten von Parametern mit unterschiedlichem Zweck und Aussehen, die in den folgenden Abschnitten beschrieben werden. Der einzige Wertetyp, der von Parametern unterstützt wird, ist die Fließkommazahl; es gibt in G-Code keine String-, Boolean- oder Integer-Typen wie in anderen Programmiersprachen. Logische Ausdrücke können jedoch mit gcode formuliert

werden: Boolesche Operatoren (*AND*, *OR*, *XOR*, und die Vergleichsoperatoren *EQ*,*NE*,*GT*,*GE*,*LT*,*LE*), sowie die *MOD*, *ROUND*, *FUP* und *FIX* Operatoren unterstützen Ganzzahlarithmetik.

Die Parameter unterscheiden sich in Syntax, Umfang, Verhalten, wenn sie noch nicht initialisiert sind, Modus, Persistenz und Verwendungszweck.

Syntax

Es gibt drei Arten der syntaktischen Erscheinung:

- *nummeriert* - #4711
- *benannt local* - #<lokaler Wert>
- *benannt global* - #<_globalvalue>

Geltungsbereich (engl. scope)

Der Geltungsbereich eines Parameters ist entweder global oder lokal innerhalb eines Unterprogramms. Unterprogramm-Parameter und lokale benannte Variablen haben einen lokalen Geltungsbereich. Globale benannte Parameter und nummerierte Parameter ab der Nummer 31 haben einen globalen Geltungsbereich. RS274/NGC verwendet *lexical scoping* - in einer Subroutine sind nur die darin definierten lokalen Variablen und alle globalen Variablen sichtbar. Die lokalen Variablen einer aufrufenden Prozedur sind in einer aufgerufenen Prozedur nicht sichtbar.

Verhalten nicht initialisierter Parameter

- Nicht initialisierte globale Parameter und nicht verwendete Unterprogrammparameter geben den Wert Null zurück, wenn sie in einem Ausdruck verwendet werden.
- Uninitialisierte benannte Parameter signalisieren einen Fehler, wenn sie in einem Ausdruck verwendet werden.

Modus

Die meisten Parameter sind schreib- und lesbar und können innerhalb einer Zuweisungsanweisung zugewiesen werden. Bei vielen vordefinierten Parametern ist dies jedoch nicht sinnvoll, daher sind sie schreibgeschützt - sie können in Ausdrücken erscheinen, aber nicht auf der linken Seite einer Zuweisungsanweisung.

Persistenz

Wenn LinuxCNC heruntergefahren wird, verlieren die flüchtigen Parameter ihre Werte. Alle Parameter mit Ausnahme der nummerierten Parameter im aktuellen persistenten Bereich [1] sind flüchtig. Persistente Parameter werden in der .var-Datei gespeichert und auf ihre vorherigen Werte zurückgesetzt, wenn LinuxCNC erneut gestartet wird. Flüchtige nummerierte Parameter werden auf Null zurückgesetzt.

Verwendungszweck

- Benutzer-Parameter - nummerierte Parameter im Bereich 31..5000 und benannte globale und lokale Parameter mit Ausnahme der vordefinierten Parameter. Diese sind für die allgemeine Speicherung von Fließkommawerten, wie Zwischenergebnisse, Flags usw., während der Programmausführung verfügbar. Sie können gelesen und geschrieben werden (ihnen kann ein Wert zugewiesen werden).
- Unterprogramm-Parameter - diese werden verwendet, um die aktuellen Parameter zu speichern,

die an ein Unterprogramm übergeben werden.

- Nummerierte Parameter - die meisten davon werden verwendet, um auf Offsets von Koordinatensystemen zuzugreifen.
- System-Parameter - werden verwendet, um die aktuell laufende Version zu ermitteln. Sie sind schreibgeschützt.

Nummerierte Parameter

Ein nummerierter Parameter ist das Doppelkreuz-Zeichen # (auch hash oder pound), gefolgt von einer ganzen Zahl zwischen 1 und (derzeit) 5602 [2]. Der Parameter wird durch diese Ganzzahl referenziert, und sein Wert ist die Zahl, die im Parameter gespeichert ist.

Mit dem =-Operator wird ein Wert in einem Parameter gespeichert, zum Beispiel:

```
#3 = 15 (Parameter 3 auf 15 setzen)
```

Eine Parametereinstellung wird erst dann wirksam, wenn alle Parameterwerte in der gleichen Zeile gefunden worden sind. Wenn beispielsweise der Parameter 3 zuvor auf 15 eingestellt wurde und die Zeile *#3=6 G1 X#3* interpretiert wird, erfolgt eine gerade Bewegung zu einem Punkt, an dem X gleich 15 ist, und der Wert von Parameter 3 wird 6 sein.

Das Zeichen # hat Vorrang vor anderen Operationen, so dass z. B. *\#1+2* die Zahl bedeutet, die sich durch Addition von 2 zum Wert von Parameter 1 ergibt, und nicht den Wert in Parameter 3. Natürlich bedeutet *\#[1+2]* den in Parameter 3 gefundenen Wert. Das #-Zeichen kann wiederholt werden; zum Beispiel bedeutet *##2* den Wert des Parameters, dessen Index der (ganzzahlige) Wert von Parameter 2 ist.

- *31-5000* - G-Code Benutzerparameter. Diese Parameter sind global in der G-Codedatei und für die allgemeine Verwendung verfügbar. Flüchtig.
- *5021-5029* - Current absolute machine position (G53) for X, Y, Z, A, B, C, U, V & W, in the current program units. Includes no coordinate system, G92/G52 or tool length offset. Same as `#<_abs_x>` through `#<_abs_w>`. Read-only, volatile.
- *5061-5069* - Koordinaten eines G38 Sondenergebnisses (X, Y, Z, A, B, C, U, V & W). Koordinaten befinden sich in dem Koordinatensystem, in dem die G38 stattfand. Flüchtig.
- *5070* - G38 Prüfpunktergebnis: 1 bei Erfolg, 0, wenn Prüfpunkt nicht geschlossen werden konnte. Verwendet mit G38.3 und G38.5. Flüchtig.
- *5081-5089* - Tool length offset currently applied to motion for X, Y, Z, A, B, C, U, V & W, in the current program units. Set by `G43`/`G43.1`/`G43.2`, and 0 when `G49` is in effect. These report the offset actually in effect, which differs from the loaded tool's stored offset in *5401-5409*. Volatile.
- *5161-5169* - "G28" Home für X, Y, Z, A, B, C, U, V & W. Persistent.
- *5181-5189* - "G30" Home für X, Y, Z, A, B, C, U, V & W. Persistent.
- *5210* - 1, wenn "G52"- oder "G92"-Offset derzeit angewendet wird, sonst 0. Standardmäßig persistent, flüchtig wenn *DISABLE_G92_PERSISTENCE = 1* im Abschnitt *[RS274NGC]* der INI-Datei gesetzt.
- *5211-5219* - Gemeinsamer "G52" und "G92" Offset für X, Y, Z, A, B, C, U, V & W. Standardmäßig flüchtig; persistent, wenn *DISABLE_G92_PERSISTENCE = 1* im Abschnitt *[RS274NGC]* der INI-Datei.

- *5220* - Koordinatensystem Nummer 1 - 9 für G54 - G59.3. Persistent.
- *5221-5230* - Koordinatensystem 1, G54 für X, Y, Z, A, B, C, U, V, W & R. R bezeichnet den XY-Drehwinkel um die Z-Achse. Persistent.
- *5241-5250* - Koordinatensystem 2, G55 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5261-5270* - Koordinatensystem 3, G56 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5281-5290* - Koordinatensystem 4, G57 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5301-5310* - Koordinatensystem 5, G58 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5321-5330* - Koordinatensystem 6, G59 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5341-5350* - Koordinatensystem 7, G59.1 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5361-5370* - Koordinatensystem 8, G59.2 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5381-5390* - Koordinatensystem 9, G59.3 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5399* - Ergebnis von M66 - Prüfen oder auf Eingabe warten. Flüchtig.
- *5400* - Werkzeugnummer. Flüchtig.
- *5401-5409* - The loaded tool's stored offset for X, Y, Z, A, B, C, U, V & W. Refreshed from the tool table on tool change (`M6`), at startup, and by `G10 L1`/`L10`/`L11`. These are not affected by `G43`/`G43.1`/`G43.2` or `G49`; for the offset actually applied to motion see *5081-5089*. Volatile.
- *5410* - Werkzeugdurchmesser. Flüchtig.
- *5411* - Werkzeug-Frontwinkel. Flüchtig.
- *5412* - Werkzeug-Rückenwinkel (engl. back angle). Flüchtig.
- *5413* - Werkzeugausrichtung. Flüchtig.
- *5420-5428* - Aktuelle relative Position im aktiven Koordinatensystem inklusive aller Offsets und in den aktuellen Programmeinheiten für X, Y, Z, A, B, C, U, V & W, flüchtig.
- *5599* - Flag zur Steuerung der Ausgabe von (DEBUG,)-Anweisungen. 1=Ausgabe, 0=keine Ausgabe; default=1. Flüchtig.

Persistenz nummerierter Parameter

Die Werte der Parameter im persistenten Bereich werden über die Zeit beibehalten, auch wenn das Bearbeitungszentrum ausgeschaltet ist. LinuxCNC verwendet eine Parameterdatei, um die Persistenz zu gewährleisten. Sie wird vom Interpreter verwaltet. Der Interpreter liest die Datei, wenn er startet, und schreibt die Datei, wenn er beendet wird.

Das Format einer Parameter-Datei ist in Tabelle Parameter-Datei-Format dargestellt.

Der Interpreter erwartet, dass die Datei zwei Spalten enthält. Er überspringt alle Zeilen, die nicht genau zwei numerische Werte enthalten. In der ersten Spalte wird ein Integer-Wert erwartet (die Nummer des Parameters). Die zweite Spalte enthält eine Fließkommazahl (der letzte Wert dieses Parameters). Der Wert wird im Interpreter als doppelt genaue Fließkommazahl dargestellt, aber ein Dezimalpunkt ist in der Datei nicht erforderlich.

Parameter im benutzerdefinierten Bereich (31-5000) können in diese Datei eingefügt werden. Solche Parameter werden vom Interpreter gelesen und in die Datei geschrieben, wenn er beendet wird.

Fehlende Parameter im persistenten Bereich werden auf Null initialisiert und beim nächsten Speichervorgang mit ihren aktuellen Werten geschrieben.

Die Parameternummern müssen in aufsteigender Reihenfolge angeordnet sein. Wenn sie nicht in aufsteigender Reihenfolge angeordnet sind, wird ein Fehler "Parameterdatei nicht in Ordnung" gemeldet.

Die Originaldatei wird als Sicherungsdatei gespeichert, wenn die neue Datei geschrieben wird.

Table 82. Parameter-Dateiformat

Parameter-Nummer	Parameter-Wert
5161	0.0
5162	0.0

Unterprogramm Codes und Parameter

Subroutine-Codes oder o-Codes (manchmal auch o-words genannt), sorgen für Logik und Flusssteuerung in NGC-Programmen (wie in if-else-Logik). Sie werden Subroutine-Codes genannt, weil sie auch Subroutinen (wie in sub-endsub) bilden können.

Siehe Kapitel über o-Codes.

NOTE Werden o-Codes zur Bildung von Subroutinen verwendet, so können o-Codes auch solche Subroutinen aufrufen und bis zu 30 Parameter angeben, die der Subroutine lokal zur Verfügung stehen und flüchtig sind. (Siehe den Abschnitt zu o-Codes für eine vollständigere Beschreibung und Beispiele.)

NOTE Während sowohl der untere als auch der obere Fall o- gültig sind, verwendet die beste Praxis den unteren Fall "o-", weil sie 0 (Null) und O (großes o) hilft zu unterscheiden.

Benannte Parameter

Benannte Parameter funktionieren wie nummerierte Parameter, sind aber einfacher zu lesen. Alle Parameternamen werden in Kleinbuchstaben umgewandelt und Leerzeichen und Tabulatoren werden entfernt, so dass sich *<param> und<P a R am >* auf denselben Parameter beziehen. Benannte Parameter müssen mit < >-Zeichen umschlossen werden.

#<benannter Parameter> ist ein lokaler benannter Parameter. Standardmäßig ist ein benannter Parameter lokal in dem Bereich, in dem er zugewiesen ist. Sie können nicht auf einen lokalen Parameter außerhalb des Unterprogramms zugreifen. Das bedeutet, dass zwei Unterprogramme die gleichen Parameternamen verwenden können, ohne dass die Gefahr besteht, dass ein Unterprogramm die Werte in einem anderen überschreibt.

#<_globaler benannter Parameter> ist ein globaler benannter Parameter. Sie sind von aufgerufenen Unterprogrammen aus zugänglich und können Werte innerhalb von Unterprogrammen setzen, die für den Aufrufer zugänglich sind. Was den Anwendungsbereich betrifft, verhalten sie sich wie normale

numerische Parameter. Sie werden nicht in Dateien gespeichert.

Beispiele:

Deklaration einer benannten globalen Variablen

```
#<_endmill_dia> = 0.049
```

Verweis auf eine zuvor deklarierte globale Variable

```
#<_endmill_rad> = [#<_endmill_dia>/2.0]
```

Gemischte literale und benannte Parameter

```
o100 call [0.0] [0.0] [#<_inside_cutout>-#<_endmill_dia>] [#<_Zcut>] [#<_feedrate>]
```

Benannte Parameter entstehen, wenn ihnen zum ersten Mal ein Wert zugewiesen wird. Lokale benannte Parameter verschwinden, wenn ihr Geltungsbereich verlassen wird: Wenn ein Unterprogramm zurückkehrt, werden alle seine lokalen Parameter gelöscht und können nicht mehr referenziert werden.

Es ist ein Fehler, einen nicht existierenden benannten Parameter innerhalb eines Ausdrucks oder auf der rechten Seite einer Zuweisung zu verwenden. Die Ausgabe des Wertes eines nicht existierenden benannten Parameters mit einer DEBUG-Anweisung - wie *(DEBUG, <kein_solcher_parameter>) - zeigt die Zeichenkette #* an.

Globale Parameter sowie lokale Parameter, die auf globaler Ebene zugewiesen werden, behalten ihren einmal zugewiesenen Wert auch nach Beendigung des Programms und haben diese Werte auch bei der erneuten Ausführung des Programms.

Die Funktion *EXISTS* prüft, ob ein bestimmter benannter Parameter existiert.

Vordefinierte benannte Parameter

Die folgenden globalen, nur lesbaren benannten Parameter sind verfügbar, um auf den internen Zustand des Interpreters und den Maschinenzustand zuzugreifen. Sie können in beliebigen Ausdrücken verwendet werden, zum Beispiel um den Programmablauf mit if-then-else-Anweisungen zu steuern. Beachten Sie, dass neue predefined named parameters einfach und ohne Änderungen am Quellcode hinzugefügt werden können.

- *#<_vmajor>* - Hauptversion des Pakets. Wenn die aktuelle Version 2.5.2 wäre, würde 2.5 zurückgegeben.
- *#<_vminor>* - Kleinere Paketversion. Wenn die aktuelle Version 2.6.2 wäre, würde es 0.2 zurückgeben.
- *#<_line>* - Sequenznummer. Wenn eine G-Code-Datei ausgeführt wird, gibt dies die aktuelle Zeilennummer zurück.
- *#<_motion_mode>* - Gibt den aktuellen Bewegungsmodus des Interpreters zurück:

Bewegungsmodus	Rückgabewert
G1	10
G2	20
G3	30
G33	330
G38.2	382
G38.3	383
G38.4	384
G38.5	385
G5.2	52
G73	730
G76	760
G80	800
G81	810
G82	820
G83	830
G84	840
G85	850
G86	860
G87	870
G88	880
G89	890

- *#<_kins_type>* - Kinematics selected by *G12.1* or *G13.1*. Returns the *P* number of the last *G12.1*, or 0 after *G13.1* or when no kinematics has been selected. See G12.1, G13.1.
- *#<_plane>* - gibt den Wert zurück, der die aktuelle Ebene bezeichnet:

Ebene	Rückgabewert
G17	170
G18	180
G19	190

Ebene	Rückgabewert
G17.1	171
G18.1	181
G19.1	191

- #<_*ccomp*> - Status der Fräserkompensation. Rückgabewerte:

Modus	Rückgabewert
G40	400
G41	410
G41.1	411
G41	410
G42	420
G42.1	421

- #<_*metric*> - Gibt 1 zurück, wenn G21 eingeschaltet ist, sonst 0.
- #<_*imperial*> - Gibt 1 zurück, wenn G20 eingeschaltet ist, sonst 0.
- #<_*absolute*> - Gibt 1 zurück, wenn G90 eingeschaltet ist, sonst 0.
- #<_*incremental*> - Gibt 1 zurück, wenn G91 eingeschaltet ist, sonst 0.
- #<_*inverse_time*> - Gibt 1 zurück, wenn der inverse Vorschubmodus (G93) eingeschaltet ist, sonst 0.
- #<_*Units_per_minute*> - Rückgabe 1, wenn der Modus Einheiten/Minute (G94) eingeschaltet ist, sonst 0.
- #<_*units_per_rev*> - Rückgabe 1, wenn der Modus Einheiten/Umdrehung (G95) eingeschaltet ist, sonst 0.
- #<_coord_system>' - Gibt eine Fließkommazahl mit dem Namen des aktuellen Koordinatensystems zurück (G54..G59.3). Wenn Sie sich beispielsweise im G55-Koordinatensystem befinden, ist der Rückgabewert 550.000000 und wenn Sie sich im G59.1-Koordinatensystem befinden, ist der Rückgabewert 591.000000.

Modus	Rückgabewert
G54	540
G55	550
G56	560
G57	570

Modus	Rückgabewert
G58	580
G59	590
G59.1	591
G59.2	592
G59.3	593

- *#<_tool_offset>* - Gibt 1 zurück, wenn Werkzeugkorrektur (G43) eingeschaltet ist, sonst 0.
- *#<_retract_r_plane>* - Rückgabe 1, wenn G98 gesetzt ist, sonst 0.
- *#<_retract_old_z>* - Rückgabe 1, wenn G99 eingeschaltet ist, sonst 0.

System-Parameter

- *#<_spindle_rpm_mode>* - Gibt 1 zurück, wenn der Spindeldrehzahlmodus (G97) eingeschaltet ist, sonst 0.
- *#<_spindle_css_mode>* - Gibt 1 zurück, wenn der Modus für konstante Schnittgeschwindigkeit (G96) eingeschaltet ist, sonst 0.
- *#<_ijk_absolute_mode>* - Gibt 1 zurück, wenn der Modus für den absoluten Bogenabstand (G90.1) eingeschaltet ist, sonst 0.
- *#<_lathe_diameter_mode>* - Gibt 1 zurück, wenn es sich um eine Drehbankkonfiguration handelt und der Durchmessermodus (G7) aktiviert ist, sonst 0.
- *#<_lathe_radius_mode>* - Gibt 1 zurück, wenn es sich um eine Drehbankkonfiguration handelt und der Radiusmodus (G8) aktiviert ist, sonst 0.
- *#<_spindle_on>* - Gibt 1 zurück, wenn die Spindel gerade läuft (M3 oder M4), sonst 0.
- *#<_spindle_cw>* - Gibt 1 zurück, wenn die Spindeldrehrichtung im Uhrzeigersinn ist (M3), sonst 0.
- *#<_mist>* - Gibt 1 zurück, wenn Nebel (M7) eingeschaltet ist.
- *#<_flut>* - Rückgabe 1, wenn Flut (M8) eingeschaltet ist.
- *#<_speed_override>* - Rückgabe 1, wenn Vorschubneufestsetzung (M48 oder M50 P1) eingeschaltet ist, sonst 0.
- *#<_feed_override>* - Gibt 1 zurück, wenn die Vorschubüberbrückung (M48 oder M51 P1) eingeschaltet ist, sonst 0.
- *#<_adaptive_feed>* - Gibt 1 zurück, wenn der adaptive Feed (M52 oder M52 P1) eingeschaltet ist, sonst 0.
- *#<_feed_hold>* - Rückgabe 1, wenn der Schalter für die Vorschubfreigabe aktiviert ist (M53 P1), sonst 0.
- *#<_feed>* - Gibt den aktuellen Wert von F zurück, nicht den tatsächlichen Vorschub.

- *#<_rpm>* - Gibt den aktuellen Wert von S zurück, nicht die tatsächliche Spindeldrehzahl.
- *#<_x>* - Gibt die aktuelle relative X-Koordinate einschließlich aller Offsets zurück. Dasselbe wie #5420. In einer Drehbank-Konfiguration wird immer der Radius zurückgegeben.
- *#<_y>* - Liefert die aktuelle relative Y-Koordinate einschließlich aller Offsets. Dasselbe wie #5421.
- *#<_z>* - Liefert die aktuelle relative Z-Koordinate einschließlich aller Offsets. Dasselbe wie #5422.
- *#<_a>* - Liefert die aktuelle relative A-Koordinate einschließlich aller Offsets. Dasselbe wie #5423.
- *#<_b>* - Gibt die aktuelle relative B-Koordinate einschließlich aller Offsets zurück. Dasselbe wie #5424.
- *#<_c>* - Gibt die aktuelle relative C-Koordinate einschließlich aller Offsets zurück. Dasselbe wie #5425.
- *#<_u>* - Liefert die aktuelle relative U-Koordinate einschließlich aller Offsets. Dasselbe wie #5426.
- *#<_v>* - Liefert die aktuelle relative V-Koordinate einschließlich aller Offsets. Dasselbe wie #5427.
- *#<_w>* - Gibt die aktuelle relative W-Koordinate einschließlich aller Offsets zurück. Dasselbe wie #5428.
- *#<_abs_x>* - Rückgabe der aktuellen absoluten X-Koordinate (G53) ohne Offsets.
- *#<_abs_y>* - Rückgabe der aktuellen absoluten Y-Koordinate (G53) ohne Offsets.
- *#<_abs_z>* - Rückgabe der aktuellen absoluten Z-Koordinate (G53) ohne Offsets.
- *#<_abs_a>* - Rückgabe der aktuellen absoluten A-Koordinate (G53) ohne Offsets.
- *#<_abs_b>* - Rückgabe der absoluten B-Koordinate (G53) ohne Offsets.
- *#<_abs_c>* - Rückgabe der aktuellen absoluten C-Koordinate (G53) ohne Offsets.
- *#<_abs_u>* - Return current absolute U coordinate (G53) including no offsets.
- *#<_abs_v>* - Return current absolute V coordinate (G53) including no offsets.
- *#<_abs_w>* - Return current absolute W coordinate (G53) including no offsets.
- *#<_aktuelles_Werkzeug>* - Rückgabe der Nummer des aktuellen Werkzeugs in der Spindel. Dasselbe wie #5400.
- *#<_current_pocket>* - Liefert den Tooldatenindex für das aktuelle Werkzeug.
- *#<_selected_tool>* - Rückgabe der Nummer des ausgewählten Werkzeugs nach einem T-Code. Voreinstellung -1.
- #<_selected_pocket>' - Gibt den tooldata-Index der ausgewählten Tasche nach einem T-Code zurück. Voreinstellung -1 (keine Tasche ausgewählt).
- *#<_value>* - Rückgabewert des letzten O-Codes *return* oder *endsub*. Standardwert 0, wenn kein Ausdruck nach *return* oder *endsub*. Wird beim Programmstart auf 0 initialisiert.
- *#<_value_returned>* - 1.0 wenn der letzte O-Code *return* oder *endsub* einen Wert zurückgegeben hat, sonst 0. Wird durch den nächsten O-Code-Aufruf gelöscht.
- *#<_task>* - 1.0 wenn die ausführende Interpreterinstanz Teil von milltask ist, sonst 0.0. Manchmal ist es notwendig, diesen Fall speziell zu behandeln, um eine korrekte Vorschau zu erhalten, z.B. beim Testen des Erfolgs einer Probe (G38.n) durch Inspektion von #5070, die im Vorschau-Interpreter (z.B.

Axis) immer fehlschlagen wird.

- *#<_call_level>* - aktuelle Verschachtelungsebene der O-Code-Prozeduren. Für die Fehlersuche.
- *#<_remap_level>* - aktuelle Ebene des Remap-Stapels. Jeder Remap in einem Block erhöht die Remap-Ebene um eins. Zur Fehlersuche.

11.4.5. HAL-Pins und INI-Werte

Wenn dies in der INI-Datei aktiviert ist, hat der G-Code Zugriff auf die Werte der INI-Datei-Einträge und HAL-Pins.

- *#<_ini[section]name>* Gibt den Wert des entsprechenden Elements in der INI-Datei zurück.

Wenn die INI-Datei zum Beispiel so aussieht:

```
[SETUP]
XPOS = 3.145
YPOS = 2.718
```

können Sie sich im G-Code auf die genannten Parameter `#<_ini[setup]xpos>` und `#<_ini[setup]ypos>` beziehen.

`EXISTS` kann verwendet werden, um das Vorhandensein einer bestimmten INI-Datei-Variable zu prüfen:

```
o100 if [EXISTS[#<_ini[setup]xpos>]]
  (debug, [setup]xpos existiert: #<_ini[setup]xpos>)
o100 else
  (debug, [setup]xpos existiert nicht)
o100 endif
```

Der Wert wird einmal aus der INI-Datei gelesen und im Interpreter zwischengespeichert. Diese Parameter sind schreibgeschützt - die Zuweisung eines Wertes führt zu einem Laufzeitfehler. Beim G-Code werden Namen nicht anhand Groß- und Kleinschreibung unterschieden - sie werden vor der Konsultation der INI-Datei in Großbuchstaben umgewandelt. Daher können INI-Einträge mit Kleinbuchstaben im Namen nicht aus dem G-Code heraus angesprochen werden.

- *#<_hal[HAL item]>* Ermöglicht es G-Code-Programmen, die Werte von HAL-Pins zu lesen. Der variable Zugriff ist schreibgeschützt, die einzige Möglichkeit, HAL-Pins von G-Code aus zu *setzen*, bleiben M62-M65, M67, M68 und benutzerdefinierte M100-M199-Codes. Beachten Sie, dass der gelesene Wert nicht in Echtzeit aktualisiert wird. Normalerweise wird der Wert zurückgegeben, der zum Zeitpunkt des Starts des G-Code-Programms an dem Pin anlag. Es ist möglich, dies zu umgehen, indem man eine Zustandssynchronisation erzwingt. Eine Möglichkeit, dies zu tun, ist ein Dummy-M66-Befehl: M66E0L0

Beispiel:

```
(debug, #<_hal[motion-controller.time]>)
```

Der Zugriff auf HAL-Elemente ist schreibgeschützt. Derzeit kann auf diese Weise nur auf HAL-Namen in

Kleinbuchstaben zugegriffen werden.

EXISTS kann verwendet werden, um das Vorhandensein eines bestimmten HAL-Elements zu testen:

```
o100 if [EXISTS[#<_hal[motion-controller.time]>]]
  (debug, [motion-controller.time] exists: #<_hal[motion-controller.time]>)
o100 else
  (debug, [motion-controller.time] does not exist)
o100 endif
```

Diese Funktion wurde durch den Wunsch nach einer stärkeren Kopplung zwischen Benutzerschnittstellenkomponenten wie GladeVCP und PyVCP motiviert, um als Parameterquelle für das Verhalten von NGC-Dateien zu fungieren. Die Alternative - durch die M6x-Pins zu gehen und sie zu verdrahten - hat einen begrenzten, nicht-mnemonischen Namensraum und ist unnötig schwerfällig, nur als UI/Interpreter-Kommunikationsmechanismus.

11.4.6. Ausdrücke (engl. expressions)

Ein Ausdruck ist eine Reihe von Zeichen, die mit einer linken Klammer *[* beginnen und mit einer ausgleichenden rechten Klammer *]* enden. Zwischen den Klammern stehen Zahlen, Parameterwerte, mathematische Operationen und andere Ausdrücke. Ein Ausdruck wird ausgewertet, um eine Zahl zu erzeugen. Die Ausdrücke in einer Zeile werden ausgewertet, wenn die Zeile gelesen wird, bevor etwas in der Zeile ausgeführt wird. Ein Beispiel für einen Ausdruck ist *[1 + acos[0] - [#3 ** [4.0/2]]]*.

11.4.7. Binäre Operatoren

Binäre Operatoren erscheinen nur innerhalb von Ausdrücken. Es gibt vier grundlegende mathematische Operationen: Addition (+), Subtraktion (-), Multiplikation (*) und Division (/). Es gibt drei logische Operationen: nicht-exklusive oder (*OR*), exklusive oder (*XOR*) und logische und (*AND*). Die achte Operation ist die Modulusoperation (*MOD*). Die neunte Operation ist die *Potenz* -Operation (**), bei der die Zahl links von der Operation mit der Potenz rechts davon erhöht wird. Die relationalen Operatoren sind Gleichheit (*EQ*), Ungleichheit (*NE*), streng größer als (*GT*), größer oder gleich (*GE*), streng kleiner als (*LT*) und kleiner als oder gleich (*LE*).

Die binären Operationen werden entsprechend ihrer Rangfolge in mehrere Gruppen unterteilt. Wenn Operationen in verschiedenen Ranggruppen aneinandergereiht werden (z. B. im Ausdruck "[2.0 / 3 * 1.5 - 5.5 / 11.0]"), sind Operationen in einer höheren Gruppe vor Operationen in einer niedrigeren Gruppe auszuführen. Wenn ein Ausdruck mehr als eine Operation aus derselben Gruppe enthält (z. B. das erste / und * im Beispiel), wird der Vorgang auf der linken Seite zuerst ausgeführt. Somit ist das Beispiel äquivalent zu: *[[[2.0 / 3] * 1.5] - [5.5 / 11.0]]* , was äquivalent zu *[1.0 - 0.5]* ist, was *0.5* ist.

Die logischen Operationen und der Modulus können mit allen reellen Zahlen durchgeführt werden, nicht nur mit ganzen Zahlen. Die Zahl Null ist gleichbedeutend mit logisch falsch, und jede Zahl ungleich Null ist gleichbedeutend mit logisch wahr.

Table 83. Vorrang der Operatoren

Operatoren	Vorrang
**	*höchste*
* / MOD	
+ -	
EQ NE GT GE LT LE	
AND OR XOR	*niedrigste*

11.4.8. Gleichheit und Gleitkommawerte

Die Sprache RS274/NGC unterstützt nur Fließkommazahlen mit doppelter Genauigkeit. Daher ist die Prüfung auf Gleichheit oder Ungleichheit zweier Fließkommazahlen von Natur aus problematisch. Der Interpreter löst dieses Problem, indem er Werte als gleich betrachtet, wenn ihre absolute Differenz kleiner als 1e-6 ist (dieser Wert ist als *TOLERANCE_EQUAL* in src/emc/rs274ngc/interp_internal.hh definiert).

11.4.9. Funktionen

Die verfügbaren Funktionen sind in der folgenden Tabelle aufgeführt. Argumente für unäre Operationen, die Winkelmaße annehmen (*COS*, *SIN*, und *TAN*), sind in Grad. Werte, die von unären Operationen zurückgegeben werden, die Winkelmaße zurückgeben (*ACOS*, *ASIN*, und *ATAN*) sind ebenfalls in Grad.

Table 84. G-Code-Funktionen

Funktionsname	Funktionsergebnis
ATAN[arg]/[arg]	Umgekehrter Tangens für alle vier Quadranten (Vier-Quadranten-Arctan)
ABS[arg]	Absoluter Wert
ACOS[arg]	Inverser Kosinus
ASIN[arg]	Inverser Sinus
COS[arg]	Kosinus
EXP[arg]	e in der angegebenen Potenz
FIX[arg]	Abrunden auf ganze Zahl
FUP[arg]	Auf Ganzzahl aufrunden
ROUND[arg]	Runden auf die nächste Ganzzahl
LN[arg]	Natürlicher Logarithmus
SIN[arg]	Sinus

Funktionsname	Funktionsergebnis
SQRT[arg]	Quadratwurzel
TAN[arg]	Tangente
EXISTS[arg]	Benannte Parameter prüfen

Die Funktion *FIX* rundet auf einer Zahlengeraden nach links (weniger positiv oder mehr negativ), so dass *FIX[2.8] =2* und *FIX[-2.8] = -3*.

Die Operation *FUP* rundet auf einer Zahlengeraden nach rechts (mehr positiv oder weniger negativ); *FUP[2.8] = 3* und *FUP[-2.8] = -2*.

Die Funktion *EXISTS* prüft, ob ein einzelner benannter Parameter vorhanden ist. Sie nimmt nur einen benannten Parameter und gibt 1 zurück, wenn er existiert, und 0, wenn er nicht existiert. Es ist ein Fehler, wenn Sie einen nummerierten Parameter oder einen Ausdruck verwenden. Hier ist ein Beispiel für die Verwendung der EXISTS-Funktion:

```
o<test> sub
o10 if [EXISTS[#<_global>]]
    (debug, _global existiert und hat den Wert #<_global>)
o10 sonst
    (debug, _global existiert nicht)
o10 endif
o<test> endsub

o<test> call
#<_global> = 4711
o<test> call
m2
```

11.4.10. Wiederholte Elemente

Eine Zeile kann eine beliebige Anzahl von G-Wörtern haben, aber zwei G-Wörter aus derselben modalen Gruppe dürfen nicht in derselben Zeile erscheinen. Weitere Informationen finden Sie im Abschnitt Modal Groups.

Eine Zeile kann null bis vier M-Wörter enthalten. Zwei M-Wörter aus der gleichen Modalgruppe dürfen nicht in der gleichen Zeile erscheinen.

Bei allen anderen Buchstaben darf eine Zeile nur ein Wort enthalten, das mit diesem Buchstaben beginnt.

Wird derselbe Parameter wiederholt in einer Zeile eingestellt, z. B. *#3=15 #3=6*, wird nur die letzte Einstellung wirksam. Es ist zwar etwas merkwürdig unnötig, aber nicht illegal, denselben Parameter zweimal in derselben Zeile zu setzen.

Wenn mehr als ein Kommentar in einer Zeile erscheint, wird nur der letzte verwendet; jeder der anderen Kommentare wird gelesen und auf sein Format geprüft, danach aber ignoriert. Es wird erwartet, dass mehr als ein Kommentar in einer Zeile sehr selten vorkommt.

11.4.11. Artikelreihenfolge

Die drei Arten von Elementen, deren Reihenfolge in einer Zeile variieren kann (wie am Anfang dieses Abschnitts angegeben), sind Wort, Parametereinstellung und Kommentar. Stellen Sie sich vor, dass diese drei Arten von Einträgen nach Typ in drei Gruppen unterteilt sind.

Die erste Gruppe (die Wörter) kann in beliebiger Reihenfolge angeordnet werden, ohne dass sich der Sinn der Zeile ändert.

Wenn die zweite Gruppe (die Parametereinstellungen) neu geordnet wird, ändert sich die Bedeutung der Zeile nicht, es sei denn, derselbe Parameter wird mehr als einmal eingestellt. In diesem Fall wird nur die letzte Einstellung des Parameters wirksam. Nachdem zum Beispiel die Zeile *#3=15 #3=6* interpretiert wurde, ist der Wert des Parameters 3 gleich 6. Wenn die Reihenfolge umgekehrt wird zu *#3=6 #3=15* und die Zeile interpretiert wird, ist der Wert von Parameter 3 15.

Wenn die dritte Gruppe (die Kommentare) mehr als einen Kommentar enthält und neu geordnet wird, dann wird nur der letzte Kommentar verwendet.

Wenn jede Gruppe in ihrer Reihenfolge beibehalten oder umgeordnet wird, ohne dass sich die Bedeutung der Zeile ändert, können die drei Gruppen in beliebiger Weise verschachtelt werden, ohne dass sich die Bedeutung der Zeile ändert. Zum Beispiel hat die Zeile *g40 g1 #3=15 (foo) #4=-7.0* fünf Elemente und bedeutet in jeder der 120 möglichen Reihenfolgen (wie *#4=-7.0 g1 #3=15 g40 (foo)*) für die fünf Elemente genau dasselbe.

11.4.12. Befehle und Maschinenmodi

Viele Befehle bewirken, dass die Steuerung von einem Modus in einen anderen wechselt, und der Modus bleibt so lange aktiv, bis er durch einen anderen Befehl implizit oder explizit geändert wird. Solche Befehle werden *modal* genannt. Zum Beispiel bleibt aach dem Einschalten des Kühlmittels dies so lange eingeschaltet, bis es explizit ausgeschaltet wird. Die G-Codes für Bewegungen sind ebenfalls modal. Wird beispielsweise ein G1-Befehl (gerade Bewegung) in einer Zeile gegeben, so wird er in der nächsten Zeile erneut ausgeführt, wenn ein oder mehrere Achsenwörter in der Zeile vorhanden sind, es sei denn, ein expliziter Befehl wird in dieser nächsten Zeile gegeben, der die Achsenwörter verwendet oder die Bewegung abbricht.

"Nicht modale" Codes wirken sich nur auf die Zeilen aus, auf denen sie vorkommen. Beispielsweise ist G4 (Verweilen) nicht modal.

11.4.13. Polarkoordinaten

Polarkoordinaten können verwendet werden, um die XY-Koordinaten einer Bewegung anzugeben. Dabei ist @n der Abstand und ^n der Winkel. Dies hat den Vorteil, dass z. B. Lochkreise sehr einfach durch Anfahren eines Punktes in der Mitte des Kreises, Einstellen des Versatzes und anschließendes Anfahren des ersten Lochs und Ausführen des Bohrzyklus erstellt werden können. Polarkoordinaten beziehen sich immer auf die aktuelle XY-Nullposition. Um die Polarkoordinaten vom Maschinennullpunkt aus zu verschieben, verwenden Sie einen Offset oder wählen Sie ein Koordinatensystem.

Im absoluten Modus beziehen sich Abstand und Winkel auf die XY-Nullposition, und der Winkel beginnt

bei 0 auf der positiven X-Achse und nimmt im Gegenuhrzeigersinn um die Z-Achse zu. Der Code G1 @1^90 ist der gleiche wie G1 Y1.

Im relativen Modus werden Abstand und Winkel ebenfalls von der XY-Nullposition aus gemessen, jedoch kumulativ. Dies kann anfangs verwirrend sein, wie dies im inkrementellen Modus funktioniert.

Wenn Sie zum Beispiel das folgende Programm haben, könnten Sie erwarten, dass es ein quadratisches Muster ist:

```
F100 G1 @.5 ^90
G91 @.5 ^90
@.5 ^90
@.5 ^90
@.5 ^90
G90 G0 X0 Y0 M2
```

Aus der folgenden Abbildung können Sie ersehen, dass die Ausgabe nicht den Erwartungen entspricht. Da wir jedes Mal 0,5 zum Abstand addiert haben, vergrößerte sich der Abstand von der XY-Nullposition mit jeder Zeile.

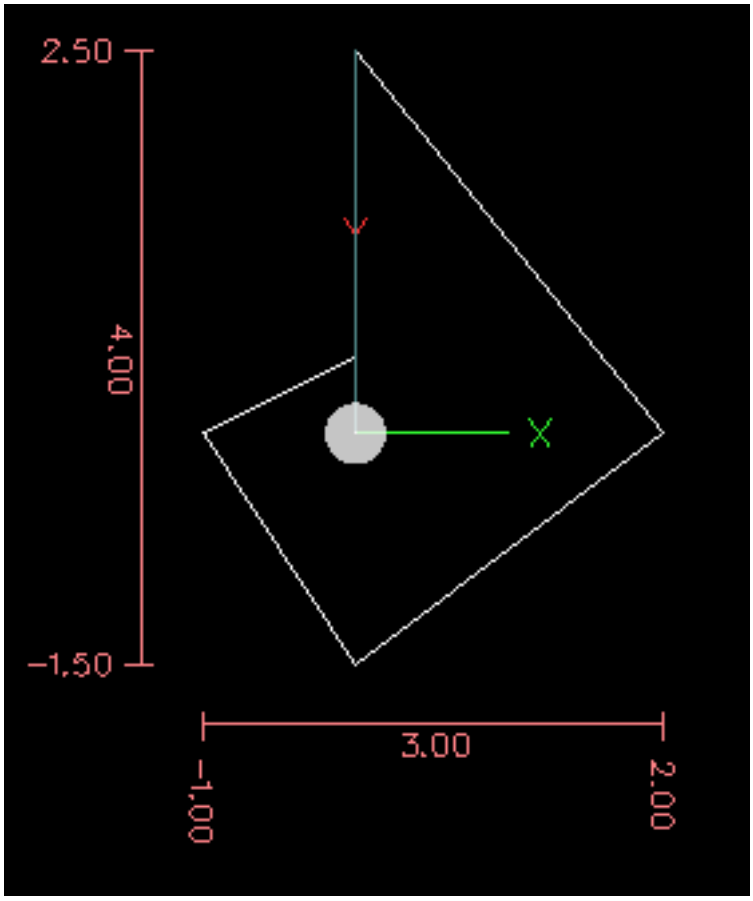


Figure 208. Polare Spirale

Der folgende Code erzeugt unser quadratisches Muster:

```
F100 G1 @.5 ^90
G91 ^90
^90
^90
```

```
^90
G90 G0 X0 Y0 M2
```

Wie Sie sehen können, ist der Endpunktabstand für jede Linie gleich, wenn Sie nur den Winkel um 90 Grad erhöhen.

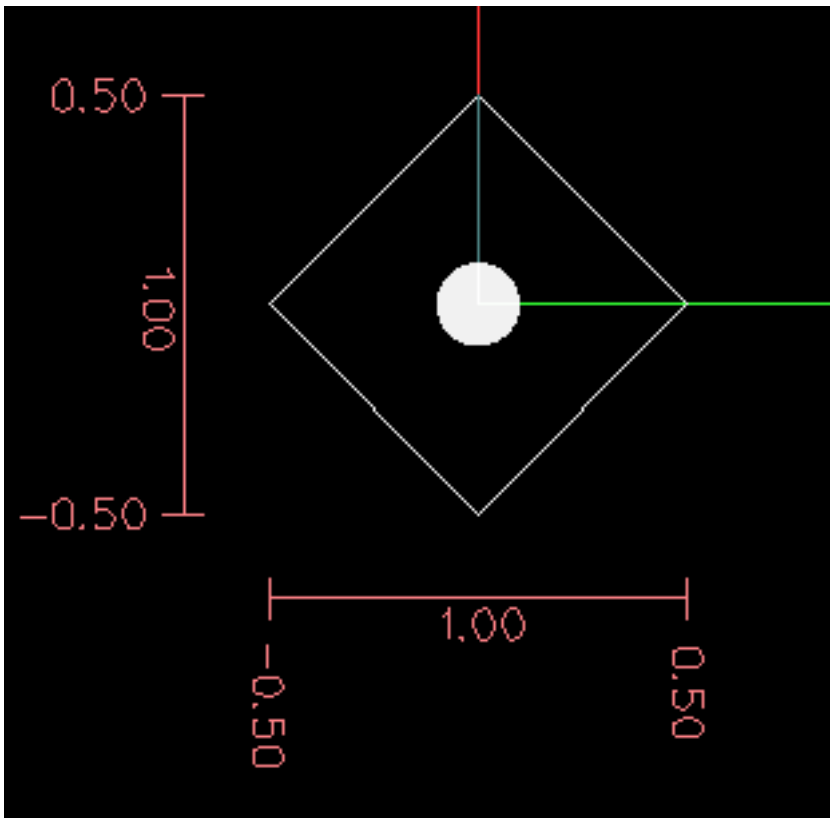


Figure 209. Polares Quadrat

Es ist ein Fehler, wenn:

- Eine inkrementelle Bewegung wird am Ursprung gestartet
- Eine Mischung aus Polar und X- oder Y-Wörtern wird verwendet

11.4.14. Modalgruppen

Modale Befehle sind in Gruppen angeordnet, die "modale Gruppen" genannt werden, und nur ein Mitglied einer modalen Gruppe kann zu einem bestimmten Zeitpunkt in Kraft sein. Im Allgemeinen enthält eine Modalgruppe Befehle, bei denen es logisch unmöglich ist, dass zwei Mitglieder gleichzeitig in Kraft sind - wie z. B. Messen in Zoll gegenüber Messen in Millimetern. Ein Bearbeitungszentrum kann sich in vielen Modi gleichzeitig befinden, wobei ein Modus aus jeder Modalgruppe in Kraft ist. Die Modalgruppen sind in der folgenden Tabelle aufgeführt.

Table 85. G-Code Modalgruppen

Bedeutung der Modalgruppe	Member-Wörter
Nicht-modale Codes (Gruppe 0)	G4, G10 G28, G28.2, G30, G52, G53, G92, G92.1, G92.2, G92.3,

Bedeutung der Modalgruppe	Member-Wörter
Bewegung (engl. motion) (Gruppe 1)	G0, G1, G2, G3, G33, G38.n, G73, G76, G80, G81 G82, G83, G84, G85, G86, G87, G88, G89
Auswahl der Ebene (Gruppe 2)	G17, G18, G19, G17.1, G18.1, G19.1
Distanzmodus (Gruppe 3)	G90, G91
Arc IJK-Distanzmodus (Gruppe 4)	G90.1, G91.1
Vorschubmodus (Gruppe 5)	G93, G94, G95
Einheiten (Gruppe 6)	G20, G21
Fräserdurchmesser-Kompensation (Gruppe 7)	G40, G41, G42, G41.1, G42.1
Werkzeuglängenausgleich (engl. tool length offset) (Gruppe 8)	G43, G43.1, G43.2, G43.4, G49
Tilted Work Plane (Group 9)	G68.2, G68.4, G69
Festzyklen Rückgabe-Modus (Gruppe 10)	G98, G99
Koordinatensystem (Gruppe 12)	G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3
Kontrollmodus (Gruppe 13)	G61, G61.1, G64
Spindeldrehzahl-Modus (Gruppe 14)	G96, G97
Drehmaschinen-Durchmessermodus (Gruppe 15)	G7, G8

Table 86. M-Code Modalgruppen

Bedeutung der Modalgruppe	Member-Wörter
Rotary Modulo Path (Group 3)	M26, M27
Anhalten (Gruppe 4)	M0, M1, M2, M30, M60
E/A (engl. I/O) Pins (Gruppe 5)	(M62-M65 digitale Ausgang), (M66 digitaler or analoger Eingang), (M67, M68 analoger Ausgang)
Werkzeugwechsel (Gruppe 6)	M6 T*n*
Spindel (Gruppe 7)	M3, M4, M5
Kühlmittel (Gruppe 8)	(M7 und M8 können beide eingeschaltet sein), M9

Bedeutung der Modalgruppe	Member-Wörter
Neufestsetzungsschalter (engl. override switches) (Gruppe 9)	M48, M49
Benutzerdefiniert (Gruppe 10)	M100-M199

Bei mehreren modalen Gruppen muss ein Mitglied der Gruppe in Kraft sein, wenn ein Bearbeitungszentrum bereit ist, Befehle anzunehmen. Für diese modalen Gruppen gibt es Standardeinstellungen. Wenn das Bearbeitungszentrum eingeschaltet oder anderweitig neu initialisiert wird, werden die Standardwerte automatisch übernommen.

Gruppe 1, die erste Gruppe auf der Tabelle, ist eine Gruppe von G-Codes für Bewegung. Einer von ihnen ist immer in Kraft. Dieser wird als der aktuelle Bewegungsmodus bezeichnet.

Es ist ein Fehler, einen G-Code der Gruppe 1 und einen G-Code der Gruppe 0 auf dieselbe Zeile zu setzen, wenn beide Achsenwörter verwenden. Wenn ein G-Code der Gruppe 1, der Achsenwörter verwendet, implizit auf einer Zeile in Kraft ist (weil er auf einer früheren Zeile aktiviert wurde) und ein G-Code der Gruppe 0, der Achsenwörter verwendet, auf der Zeile erscheint, wird die Aktivität des G-Codes der Gruppe 1 für diese Zeile ausgesetzt. Die Achsenwort-verwendenden G-Codes der Gruppe 0 sind G10, G28, G30, G52 und G92.

Es ist ein Fehler, irgendwelche nicht zusammenhängende Wörter in eine Zeile mit *O*--Flusssteuerung aufzunehmen.

11.4.15. Kommentare

Kommentare sind rein informativ und haben keinen Einfluss auf das Verhalten der Maschine.

Kommentare können zu Zeilen von G-Code hinzugefügt werden, um die Absicht des Programmierers zu verdeutlichen. Kommentare können in einer Zeile mit Klammern () oder für den Rest der Zeile mit einem Semikolon eingebettet werden. Das Semikolon wird nicht als Beginn eines Kommentars behandelt, wenn es in Klammern eingeschlossen ist.

Kommentare können zwischen Wörtern stehen, aber nicht zwischen Wörtern und dem entsprechenden Parameter. So ist *S100(set speed)F200(feed)* in Ordnung, *S(speed)100F(feed)* hingegen nicht.

Hier ist ein Beispiel für ein kommentiertes Programm:

```
G0 (Schnellstart) X1 Y1
G0 X1 Y1 (Schnellstart; aber das Kühlmittel nicht vergessen)
M2 ; Ende des Programms.
```

Es gibt mehrere *aktive* Kommentare, die wie Kommentare aussehen, aber eine Aktion auslösen, wie z.B. *(debug,..)* oder *(print,..)*. Wenn es mehrere Kommentare in einer Zeile gibt, wird nur der letzte Kommentar nach diesen Regeln interpretiert. Daher wird ein normaler Kommentar, der auf einen aktiven Kommentar folgt, den aktiven Kommentar deaktivieren. Zum Beispiel wird *(foo) (debug,#1)* den Wert des Parameters *#1* ausgeben, *(debug,#1)(foo)* jedoch nicht.

Ein Kommentar, der durch ein Semikolon eingeleitet wird, ist per Definition der letzte Kommentar in dieser Zeile und wird immer als aktive Kommentarsyntax interpretiert.

NOTE	Inline-Kommentare zu O-Codes sollten nicht verwendet werden, siehe den Abschnitt O-Code Kommentare für weitere Informationen.

11.4.16. Nachrichten

- *(MSG,)* - zeigt eine Meldung an, wenn *MSG* nach der linken Klammer und vor einem anderen Druckzeichen erscheint. Varianten von *MSG*, die Leerzeichen und Kleinbuchstaben enthalten, sind zulässig. Der Rest der Zeichen vor der rechten Klammer wird als Nachricht betrachtet. Meldungen sollten auf dem Meldungsanzeigegerät der Benutzeroberfläche angezeigt werden, falls vorhanden.

Beispiel für eine Nachricht

```
(MSG, Dies ist eine Nachricht)
```

11.4.17. Sensor Protokollierung (engl. probe logging)

- *(PROBEOPEN dateiname.txt)* - öffnet dateiname.txt und speichert darin die 9-stellige Koordinate, bestehend aus XYZABCUVW, jeder erfolgreichen geraden Probe.
- *(PROBECLOSE)* - schließt die geöffnete Probelog-Datei.

Weitere Informationen zur Sondierung finden Sie im Abschnitt G38.

11.4.18. Protokollierung (engl. logging)

- *(LOGOPEN,Dateiname.txt)* - öffnet die genannte Protokolldatei. Wenn die Datei bereits existiert, wird sie abgeschnitten.
- *(LOGAPPEND,Dateiname)* - öffnet die genannte Protokolldatei. Wenn die Datei bereits existiert, werden die Daten angehängt.
- *(LOGCLOSE)* - schließt eine geöffnete Protokolldatei.
- *(LOG,)* - alles, was über das , hinausgeht, wird in die Protokolldatei geschrieben, wenn sie geöffnet ist. Unterstützt die Erweiterung von Parametern wie unten beschrieben.

Beispiele für die Protokollierung finden Sie in den Beispiel-G-Code-Dateien *nc_files/examples/smartprobe.ngc* und *nc_files/ngcgui_lib/rectange_probe.ngc*.

11.4.19. Abbruch Nachrichten (engl. abort messages)

- *(ABORT,)* - zeigt eine Meldung wie *(MSG,)* an, mit dem Zusatz einer besonderen Behandlung von Kommentarparametern, wie unten beschrieben, und den laufenden Prozess abbrechend.

11.4.20. Debug-Meldungen

- *(DEBUG,)* - zeigt eine Meldung wie *(MSG,)* an, mit dem Zusatz einer besonderen Behandlung von Kommentarparametern, wie unten beschrieben.

11.4.21. Nachrichten ausgeben

- *(PRINT,)* - Meldungen werden auf *stderr* ausgegeben, wobei Kommentarparameter wie unten beschrieben besonders behandelt werden.

11.4.22. Kommentar Parameter

In den Kommentaren DEBUG, PRINT und LOG werden die Werte der Parameter in der Meldung erweitert.

Zum Beispiel: um eine benannte globale Variable auf stderr (das Standard-Konsolenfenster) auszugeben.

Parameter Beispiel

```
(print,Endfräserdurchmesser = #<_endmill_dia>)
(print,Wert der Variablen 123 ist: #123)
```

Innerhalb der oben genannten Arten von Kommentaren werden Sequenzen wie "#123" durch den Wert des Parameters 123 ersetzt. Sequenzen wie "\#<benannter Parameter>" werden durch den Wert des benannten Parameters ersetzt. Bei benannten Parametern wird das Leerzeichen entfernt. So wird *#<Benannter Parameter>* in *#<Benannter Parameter>* umgewandelt.

Parameternummern können formatiert werden, z.B.:

```
(DEBUG, Wert = %d#<some_value>)
```

gibt den Wert gerundet auf eine ganze Zahl aus.

- %lf ist Standard, wenn keine Formatierungszeichenfolge vorhanden ist.
- %d = keine Dezimalstellen
- %f = vier Dezimalstellen
- %.xf = x (0-9) explizite Angabe der Anzahl an Dezimalstellen

Die Formatierung wird für alle Parameter in derselben Zeile durchgeführt, sofern sie nicht geändert werden, d.h. mehrere Formatierungen in einer Zeile sind zulässig.

Die Formatierungszeichenfolge muss nicht direkt neben dem Parameter stehen.

Wird die Formatierungszeichenfolge mit dem falschen Muster erstellt, so wird sie als Zeichen gedruckt.

11.4.23. Workpiece

A *(WORKPIECE,...)* comment declares the stock the program is cut from. The G-code preview draws it as a wireframe outline around the toolpath, and, where the GUI's workpiece opacity is set above zero, as a translucent solid with that outline over it. It has no effect on machine behaviour, on the program extents, or on zoom-to-fit.

Three shapes are understood:

```
(WORKPIECE,BOX,XMIN=0,YMIN=0,ZMIN=-40,XMAX=100,YMAX=100,ZMAX=0)
(WORKPIECE,CYLINDER,AXIS=Z,X=0,Y=0,ZMIN=-40,ZMAX=0,OD=80)
(WORKPIECE,TUBE,AXIS=Z,X=0,Y=0,ZMIN=-40,ZMAX=0,OD=80,ID=40)
```

Keys may appear in any order and are case-insensitive, as are the shape names.

Table 87. Keys

Key	Shapes	Bedeutung
XMIN YMIN ZMIN XMAX YMAX ZMAX	BOX	The two opposite corners. All six are required, and no *MIN* may exceed its *MAX*.
AXIS	CYLINDER, TUBE	The axis of revolution, *X*, *Y* or *Z*. Optional, default *Z*.
<axis>MIN <axis>MAX	CYLINDER, TUBE	The extent along *AXIS* - so *ZMIN*/*ZMAX* for *AXIS=Z*. Both required.
the other two axis letters	CYLINDER, TUBE	Where the axis sits in the other two directions - *X* and *Y* for *AXIS=Z*. Optional, default 0.
OD	CYLINDER, TUBE	The outside diameter; required, and always a diameter, never a radius, regardless of lathe diameter mode (G7).
ID	TUBE	The bore diameter; required, and must be greater than zero and less than *OD*.
UNITS	all	*MM* or *INCH*. Optional; without it the values are read in the units modal at the comment (G20/G21).

Several *(WORKPIECE,...)* comments are additive: each one adds a piece of stock, so a fixture holding several parts can be described with one comment per piece.

The coordinates are read in the coordinate system active where the comment appears - the G92 offset,

the G5x XY rotation and the G5x offset in force at that line, exactly as for a move endpoint on the same line. There is no key to name a coordinate system; a post processor must therefore emit the comment *after* the work offset it belongs to:

```
G54
(WORKPIECE,BOX,XMIN=0,YMIN=0,ZMIN=-40,XMAX=100,YMAX=100,ZMAX=0)
```

Unknown keys are ignored, so a comment written for a later version of LinuxCNC still draws what this one understands. Anything else malformed - a missing required key, a value that is not a number, an unrecognised shape - is ignored with a warning, and the rest of the file is interpreted normally. A workpiece declared between *(AXIS,hide)* and *(AXIS,show)* is still drawn: those comments suppress moves, and the stock is not a move.

11.4.24. Datei Anforderungen

Eine G-Code-Datei muss eine oder mehrere Zeilen G-Code enthalten und mit einem Programmende abgeschlossen werden. Jeder G-Code nach dem Programmende wird nicht ausgewertet.

Wenn kein Programmendcode verwendet wird, sollte der auszuführende Code von ein Paar von Prozentzeichen % begrenzt werden. Die ersten Prozentzeichen stehen in der ersten Zeile der Datei, gefolgt von einer oder mehreren Zeilen G-Code und einem zweiten Prozentzeichen. Jeder Code nach dem zweiten Prozentzeichen wird nicht ausgewertet.

WARNING Die Verwendung von % zum Umschließen einer G-Code-Datei bewirkt nicht dasselbe wie die Verwendung eines Programmendes. Die Maschine befindet sich in dem Zustand, in dem das Programm sie mit % verlassen hat, die Spindel und das Kühlmittel können noch eingeschaltet sein und Dinge wie G90/91 sind noch so, wie sie im letzten Programm eingestellt waren. Wenn Sie keine korrekte Präambel verwenden, könnte das nächste Programm in einem gefährlichen Zustand starten.

NOTE Die Datei muss mit einem Texteditor wie Gedit erstellt werden und nicht mit einem Textverarbeitungsprogramm wie Open Office Word Processor.

11.4.25. Dateigröße

Der Interpreter und die Task sind sorgfältig geschrieben, so dass die einzige Grenze für die Größe des Teilprogramms die Festplattenkapazität ist. Die TkLinuxCNC- und Axis-Schnittstelle laden beide den Programmtext, um ihn dem Benutzer anzuzeigen, so dass der RAM-Speicher ein begrenzender Faktor wird. Da in Axis die Vorschau standardmäßig gezeichnet wird, ist die Zeit, die für das Neuzeichnen benötigt wird, auch eine praktische Grenze für die Programmgröße. Die Vorschau kann in Axis ausgeschaltet werden, um das Laden großer Teileprogramme zu beschleunigen. In Axis können Teile der Vorschau mit dem Kommentar preview control ausgeschaltet werden.

11.4.26. G-code Ausführungsreihenfolge

Die Reihenfolge der Ausführung der Posten in einer Zeile wird nicht durch die Position der einzelnen

Posten in der Zeile bestimmt, sondern durch die folgende Liste:

- O-Code-Befehle (optional gefolgt von einem Kommentar, aber keine anderen Wörter in der gleichen Zeile erlaubt)
- Kommentar (einschließlich Nachricht)
- Vorschubmodus einstellen (G93, G94).
- Vorschubgeschwindigkeit (F) einstellen.
- Spindeldrehzahl (S) einstellen.
- Werkzeug auswählen (T).
- HAL-Pin-E/A (M62-M68).
- Werkzeug wechseln (M6) und Werkzeugnummer einstellen (M61).
- Spindel ein- oder ausschalten (M3, M4, M5).
- Status speichern (M70, M73), Wiederherstellung des Status (M72), Status ungültig machen (M71).
- Kühlmittel ein- oder ausschalten (M7, M8, M9).
- Aktivieren oder Deaktivieren von Neufestsetzungen (engl. overrides) (M48, M49,M50,M51,M52,M53).
- Benutzerdefinierte Befehle (M100-M199).
- Verweilen (engl. dwell) (G4).
- Aktive Ebene einstellen (G17, G18, G19).
- Längeneinheiten einstellen (G20, G21).
- Fräserradiuskorrektur ein oder aus (G40, G41, G42)
- Fräserlängenkorrektur ein oder aus (G43, G49)
- Auswahl des Koordinatensystems (G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3).
- Bahnsteuerungsmodus einstellen (G61, G61.1, G64)
- Abstandsmodus einstellen (G90, G91).
- Rückzugsmodus einstellen (G98, G99).
- Referenzpunkt anfahren (G28, G30) oder Koordinatensystemdaten ändern (G10) oder Achsenoffsets einstellen (G52, G92, G92.1, G92.2, G92.3).
- Bewegung ausführen (G0 bis G3, G33, G38.n, G73, G76, G80 bis G89), eventuell modifiziert durch G53.
- Stopp (M0, M1, M2, M30, M60).

11.4.27. G-Code beste Praktiken

Verwenden einer angemessenen Dezimalgenauigkeit

Verwenden Sie mindestens 3 Nachkommastellen, wenn Sie in Millimetern fräsen, und mindestens 4 Nachkommastellen, wenn Sie in Zoll fräsen.

Insbesondere werden Toleranzprüfungen der Bögen für .001 und .0001 entsprechend den aktiven Einheiten durchgeführt.

Lehrzeichen konsistent nutzen

G-Code ist am besten lesbar, wenn vor den Wörtern mindestens ein Leerzeichen steht. Es ist zwar erlaubt, Leerzeichen in der Mitte von Zahlen einzufügen, aber es gibt keinen Grund, dies zu tun.

Bögen im Zentrum-Format verwenden

Bögen im Zentrum-Format (engl. center format) (die „I-J-K-“ anstelle von „R-“ verwenden) verhalten sich konsistenter als Bögen im R-Format, insbesondere bei eingeschlossenen Winkeln nahe 180 oder 360 Grad.

Verwenden Sie eine Präambel für modale Gruppen

Wenn die korrekte Ausführung Ihres Programms von Modaleinstellungen abhängt, sollten Sie diese zu Beginn des Werkstück-Programms festlegen. Modi können von früheren Programmen und von den MDI-Befehlen übernommen werden.

Beispiel einer Präambel für eine Fräse

```
G17 G20 G40 G49 G54 G80 G90 G94
```

G17 XY-Ebene verwenden, G20 Zoll-Modus, G40 Durchmesserkompensation aufheben, G49 Längenversatz aufheben, G54 Koordinatensystem 1 verwenden, G80 Festzyklen aufheben, G90 Absolutweg-Modus, G94 Vorschub/Minuten-Modus.

Die vielleicht wichtigste modale Einstellung ist die Abstandseinheit - wenn Sie G20 oder G21 nicht einbeziehen, fräsen verschiedene Maschinen das Programm in unterschiedlichen Maßstäben. Andere Einstellungen, wie der Rücklaufmodus bei Festzyklen, können ebenfalls wichtig sein.

Nicht zu viele Dinge in eine Zeile packen

Ignorieren Sie alles, was in Abschnitt Reihenfolge der Ausführung steht, und schreiben Sie stattdessen keine Codezeile, die auch nur ein bisschen zweideutig ist.

Einen Parameter nicht in der gleichen Zeile setzen und verwenden

Verwenden und setzen Sie einen Parameter nicht in der gleichen Zeile, auch wenn die Semantik klar definiert ist. Die Aktualisierung einer Variablen auf einen neuen Wert, z. B. *#1=[#1+#2]*, ist in Ordnung.

Verwenden Sie keine Zeilennummern

Zeilennummern bieten keine Vorteile. Wenn Zeilennummern in Fehlermeldungen angegeben werden, beziehen sich die Nummern auf die Zeilennummer in der Datei, nicht auf den N-Wort-Wert.

Wenn mehrere Koordinatensysteme verschoben werden

Erwägen Sie die Verwendung des umgekehrten Zeit-Geschwindigkeits-Modus (inverse time speed mode).

Da die Bedeutung eines "F"-Wortes in Metern pro Minute je nach Art der zu bewegenden Achse variiert und die Menge des abgetragenen Materials nicht nur von der Vorschubgeschwindigkeit abhängt, kann es einfacher sein, G93, die inverse Geschwindigkeit der Zeit, zu verwenden, um den Abtrag des gewünschten Materials zu erreichen.

11.4.28. Lineare und rotierende Achsen

Da die Bedeutung eines F-Wortes im Vorschub-pro-Minute-Modus davon abhängt, welche Achsen zu bewegen sind, und da die Menge des abgetragenen Materials nicht nur von der Vorschubgeschwindigkeit abhängt, kann es einfacher sein, den G93-Modus für den inversen Zeitvorschub zu verwenden, um die gewünschte Materialabtragsrate zu erreichen.

11.4.29. Häufige Fehlermeldungen

- G-Code außerhalb des Bereichs' - Ein G-Code größer als G99 wurde verwendet, der Umfang der G-Codes in LinuxCNC ist 0 - 99. Nicht jede Zahl zwischen 0 und 99 ist ein gültiger G-Code.
- Unbekannter G-Code verwendet' - Es wurde ein G-Code verwendet, der nicht Teil der LinuxCNC G-Code Sprache ist.
- *i,j,k Wort ohne Gx, um es zu verwenden* - i, j und k Wörter müssen in der gleichen Zeile wie der G-Code verwendet werden.
- "Achsenwerte können nicht ohne einen G-Code verwendet werden, der sie verwendet" - Achsenwerte können nicht in einer Zeile verwendet werden, ohne dass entweder ein modaler G-Code oder ein G-Code in derselben Zeile wirksam ist.
- *Datei endete ohne Prozentzeichen oder Programmende* - Jede G-Code-Datei muss in einem M2 oder M30 enden oder mit dem Prozentzeichen umschlossen sein.

11.5. G-Codes

11.5.1. Konventionen

In diesem Abschnitt verwendete Konventionen

In den G-Code-Prototypen steht der Bindestrich (-) für einen realen Wert und (<>) für ein optionales Element.

Wenn "L-" in einem Prototyp geschrieben wird, so wird das "-" oft als "L-Nummer" bezeichnet, und so weiter für jeden anderen Buchstaben.

In den G-Code-Prototypen steht das Wort "Achsen" (oder noch englisch "axes") für jede Achse, die in Ihrer Konfiguration definiert ist.

Ein optionaler Wert wird wie folgt geschrieben: <*L-* >.

Ein echter Wert kann sein:

- Eine explizite Zahl, *4*
- Ein Ausdruck, *[2+2]*
- Ein Parameterwert, *#88*
- Ein unärer Funktionswert, *acos[0]*

In den meisten Fällen, wenn "Achsen"-Wörter angegeben werden (einige oder alle von "X Y Z A B C U V

W"), geben sie einen Zielpunkt an.

Die Achsennummern beziehen sich auf das derzeit aktive Koordinatensystem, es sei denn, es wird ausdrücklich als absolutes Koordinatensystem bezeichnet.

Wenn Achsenwörter optional sind, behalten ausgelassene Achsen ihren ursprünglichen Wert.

Alle Elemente in den G-Code-Prototypen, die nicht ausdrücklich als optional beschrieben werden, sind obligatorisch.

Die Werte, die auf Buchstaben folgen, werden oft als explizite Zahlen angegeben. Sofern nicht anders angegeben, können die expliziten Zahlen reelle Werte sein. Zum Beispiel könnte "G10 L2" genauso gut als "G[2*5] L[1+1]" geschrieben werden. Wäre der Wert des Parameters 100 gleich 2, würde "G10 L#100" dasselbe bedeuten.

Wenn "L-" in einem Prototyp geschrieben wird, so wird das "-" oft als "L-Nummer" bezeichnet, und so weiter für jeden anderen Buchstaben.

11.5.2.G-Code Kurzübersichts-Tabelle

Code	Beschreibung
G0	Koordinierte Bewegung im Eiltempo
G1	Koordinierte Bewegung mit Vorschubgeschwindigkeit
G2 G3	Koordinierte schraubenförmige (helikale) Bewegung mit Vorschubgeschwindigkeit
G4	Verweilen (engl. dwell)
G5	Kubischer Spline
G5.1	Quadratischer B-Spline
G5.2,G5.3	NURBS, Kontrollpunkt hinzufügen
G7	Durchmesser-Modus (Drehmaschine)
G8	Radius-Modus (Drehmaschine)
G10 L0	Werkzeug-Tabellendaten neu laden
G10 L1	Werkzeugtabelleneintrag festlegen
G10 L10	Bestimme Werkzeugtabelle, Berechnet, Werkstück
G10 L11	Bestimme Werkzeugtabelle, Berechnet, Spannmittel

Code	Beschreibung
G10 L2	Festlegung des Koordinatensystem-Ursprungs
G10 L20	Ursprungseinstellung des Koordinatensystems berechnet
G12.1 G13.1	Select Kinematics
G17 - G19.1	Ebene auswählen
G20 G21	Maßeinheiten festlegen
G28 - G28.1	Zur vordefinierten Position gehen
G28.2	Home from G-code
G30 - G30.1	Zur vordefinierten Position gehen
G33	Spindelsynchronisierte Bewegung
G33.1	Starres Gewindeschneiden
G38.2 - G38.5	Sondieren
G40	Fräserkompensation abbrechen
G41 G42	Fräserkompensation
G41.1 G42.1	Dynamische Fräserkompensation
G43	Werkzeuglängenversatz aus der Werkzeugtabelle verwenden
G43.1	Dynamischer Werkzeuglängenversatz
G43.2	Zusätzlichen Werkzeuglängenversatz anwenden
G43.4	Tool Length Offset on Primary Kinematics
G49	Werkzeuglängenversatz abbrechen
G52	Versatz des lokalen Koordinatensystems
G53	Bewegen in Maschinenkoordinaten
G54-G59.3	Koordinatensystem auswählen (1 - 9)
G61	Exakter Pfad Modus
G61.1	Exakter Stopp-Modus
G64	Bahnsteuerungsmodus mit optionaler Toleranz

Code	Beschreibung
G68.2 G68.4 G69	Tilted Work Plane
G70	Endbearbeitungszyklus der Drehmaschine
G71-G72	Schruppzyklus der Drehmaschine
G73	Bohrzyklus mit Spanbruch
G74	Linkshändiger Gewindeschneidzyklus mit Verweilzeit
G76	Gewindeschneidzyklus mit mehreren Durchgängen (Drehmaschine)
G80	Bewegungsmodi abbrechen
G81	Bohrzyklus
G82	Bohrzyklus mit Verweilzeit (engl. dwell)
G83	Bohrzyklus mit Peck
G84	Rechtsgewinde-Bohrzyklus mit Verweilzeit (engl. dwell)
G85	Bohrzyklus, keine Verweilzeit, Vorschub
G86	Bohrzyklus, Stopp, Eilgang raus
G87	Back-boring Cycle *(noch nicht implementiert)*
G88	Boring Cycle, Stop, Manual Out *(noch nicht implementiert)*
G89	Bohrzyklus, Verweilen, Vorschub Raus
G90 G91	Distanz-Modus
G90.1 G91.1	Bogenabstandsmodus (engl. Arc Distance Mode)
G92	Koordinatensystem-Versatz
G92.1 G92.2	G92-Offsets abbrechen
G92.3	G92 Offsets wiederherstellen
G93 G94 G95	Vorschub-Modi (engl. feed modes)

Code	Beschreibung
G96 G97	Spindelsteuerungsmodus, konstante Oberfläche vs. Drehzahl (IPM oder m/min vs. U/min)
G98 G99	Canned Cycle Z Rückzugsmodus

11.5.3.G0 Eilgang Bewegung

```
G0 <Achsen>
```

Für den Eilgang programmieren Sie *G0-Achsen*, wobei alle Achsenwörter optional sind. Das *G0* ist optional, wenn der aktuelle Bewegungsmodus *G0* ist. Dies führt zu einer koordinierten Bewegung zum Zielpunkt mit der maximalen Eilgeschwindigkeit (oder langsamer). *G0* wird typischerweise als Positionierbewegung verwendet.

Eilgangs-Geschwindigkeitsrate

Die Einstellung MAX_VELOCITY im Abschnitt [TRAJ] der INI-Datei definiert die maximale Eilganggeschwindigkeit. Die maximale Eilganggeschwindigkeit kann bei einer koordinierten Bewegung höher sein als die MAX_VELOCITY-Einstellung der einzelnen Achsen. Der maximale Eilgang kann langsamer sein als die MAX_VELOCITY-Einstellung in der Sektion [TRAJ], wenn eine Achsen-MAX_VELOCITY oder Trajektorienbeschränkungen ihn begrenzen.

G0 Beispiel

```
G90 (Einstellung des absoluten Abstandsmodus)
G0 X1 Y-2.3 (Schnelle lineare Bewegung von der aktuellen Position zu X1 Y-2.3)
M2 (Programm beenden)
```

- Siehe G90 & M2 für weitere Informationen.

Wenn die Fräserkompensation aktiv ist, weicht die Bewegung von der obigen ab; siehe Abschnitt Fräser-Kompensation.

Wenn *G53* in der gleichen Zeile programmiert wird, unterscheidet sich auch die Bewegung; siehe den Abschnitt G53 für weitere Informationen.

Die Bahn einer G0-Eilgangbewegung kann bei Richtungsänderungen abgerundet werden und hängt von den Trajektions-Steuerung (engl. trajectory control)-Einstellungen und der maximalen Beschleunigung der Achsen ab.

Es ist ein Fehler, wenn:

- Ein Achsenbuchstabe ohne reellen (Gleitkommazahl) Wert angegeben wird.
- Ein Achsenbuchstabe verwendet wird, der nicht konfiguriert ist.

11.5.4.G1 Lineare Bewegung

```
G1-Achsen
```

Für eine lineare (geradlinige) Bewegung mit der programmierten Vorschubgeschwindigkeit (zum Schneiden oder nicht), programmieren Sie *G1 'Achsen'*, wobei alle Achsenwörter optional sind. Das *G1* ist optional, wenn der aktuelle Bewegungsmodus *G1* ist. Dies führt zu einer koordinierten Bewegung zum Zielpunkt mit der aktuellen Vorschubgeschwindigkeit (oder langsamer).

G1 Beispiel

```
G90 (absoluter Abstandsmodus einstellen)
G1 X1.2 Y-3 F10 (lineare Bewegung mit einem Vorschub von 10 von der aktuellen Position
nach X1.2 Y-3)
Z-2.3 (lineare Bewegung mit gleichem Vorschub von der aktuellen Position nach Z-2.3)
Z1 F25 (lineare Bewegung mit einem Vorschub von 25 von der aktuellen Position nach Z1)
M2 (Programm beenden)
```

- Siehe die Abschnitte G90 & F & M2 für weitere Informationen.

Wenn die Fräserkompensation aktiv ist, weicht die Bewegung von der obigen ab; siehe Abschnitt Fräser-Kompensation.

Wenn *G53* in der gleichen Zeile programmiert wird, unterscheidet sich auch die Bewegung; siehe den Abschnitt G53 für weitere Informationen.

Es ist ein Fehler, wenn:

- Es wurde keine Vorschubgeschwindigkeit eingestellt.
- Ein Achsenbuchstabe ohne reellen (Gleitkommazahl) Wert angegeben wird.
- Ein Achsenbuchstabe verwendet wird, der nicht konfiguriert ist.

11.5.5.G2, G3 Bogenbewegung

```
G2 oder G3 Achsen Offsets (Zentrum-Format)
G2 oder G3 Achsen R- (Radius-Format)
G2 oder G3 Offsets|R- <P-> (Vollkreise)
```

Ein kreisförmiger oder spiralförmiger ("helikaler") Bogen wird entweder mit *G2* (Bogen im Uhrzeigersinn) oder *G3* (Bogen gegen den Uhrzeigersinn) mit dem aktuellen Vorschub angegeben. Die Richtung (im oder gegen den Uhrzeigersinn, engl. kurz CW oder CCW) ist vom positiven Ende der Achse aus gesehen, um welche die Kreisbewegung erfolgt.

Die Achse des Kreises oder der Spirale/Helix muss parallel zur X-, Y- oder Z-Achse des Maschinenkoordinatensystems liegen. Die Achse (bzw. die Ebene senkrecht zur Achse) wird mit *G17* (Z-Achse, XY-Ebene), *G18* (Y-Achse, XZ-Ebene) oder *G19* (X-Achse, YZ-Ebene) ausgewählt. Die Ebenen *17.1*, *18.1* und *19.1* werden derzeit nicht unterstützt. Wenn der Bogen kreisförmig ist, liegt er in einer Ebene parallel zur ausgewählten Ebene.

Um eine Helix zu programmieren, fügen Sie das Achsenwort senkrecht zur Bogenebene ein, z.B. wenn Sie in der Ebene *G17* sind, fügen Sie ein *Z*-Wort ein. Dies bewirkt, dass sich die Z-Achse während der kreisförmigen XY-Bewegung auf den programmierten Wert bewegt.

Um einen Bogen zu programmieren, der mehr als eine volle Umdrehung ergibt, verwenden Sie das Wort "P", das die Anzahl der vollen Umdrehungen plus des programmierten Bogens angibt. Das *P*-Wort muss eine ganze Zahl sein. Wird *P* nicht angegeben, verhält es sich so, als ob *P1* eingegeben wurde, d.h. es wird nur eine volle oder teilweise Umdrehung ausgeführt. Wird z. B. ein Bogen von 180 Grad mit P2 programmiert, beträgt die resultierende Bewegung 1 1/2 Umdrehungen. Für jedes P-Inkrement über 1 wird dem programmierten Bogen ein zusätzlicher Vollkreis hinzugefügt. Spiralförmige/helikale Bewegungen mit mehreren Umdrehungen werden unterstützt und ermöglichen Bewegungen, die zum Fräsen von Löchern oder Gewinden nützlich sind.

WARNING	Wenn die Steigung der Helix sehr klein ist (kleiner als die naive CAM tolerance), wird die Helix möglicherweise in eine gerade Linie umgewandelt. Bug #222

Wenn eine Codezeile einen Bogen macht und eine Drehachsenbewegung enthält, drehen sich die Drehachsen mit einer konstanten Geschwindigkeit, so dass die Drehbewegung beginnt und endet, wenn die XYZ-Bewegung beginnt und endet. Zeilen dieser Art werden fast nie programmiert.

Wenn die Fräserkompensation aktiv ist, weicht die Bewegung von der obigen ab; siehe Abschnitt Fräser-Kompensation.

Der Bogenmittelpunkt ist absolut oder relativ, wie mit G90.1 oder G91.1 festgelegt.

Für die Angabe eines Bogens sind zwei Formate zulässig: Zentrumsformat und Radiusformat.

Es ist ein Fehler, wenn:

- Es wurde keine Vorschubgeschwindigkeit eingestellt.
- Das P-Wort ist keine ganze Zahl.

Bögen durch ihr Zentrum beschrieben

Bögen mit bekannter Mitte sind genauer als Bögen im Radiusformat und werden daher bevorzugt verwendet.

Der Endpunkt des Bogens und der Abstand zum Mittelpunkt des Bogens von der aktuellen Position werden verwendet, um Bögen zu programmieren, die weniger als ein Vollkreis sind. Es ist in Ordnung, wenn der Endpunkt des Bogens mit der aktuellen Position identisch ist.

Zur Programmierung von Vollkreisen werden der Abstand zum Mittelpunkt des Bogens von der aktuellen Position und optional die Anzahl der Wiederholungen verwendet.

Bei der Programmierung von Bögen kann es zu Rundungsfehlern kommen, wenn eine Genauigkeit von weniger als 4 Dezimalstellen (0.0000) bei Zoll und weniger als 3 Dezimalstellen (0.000) bei Millimetern verwendet wird. Es wird ein Dezimalpunkt erwartet.

Inkrementeller Bogenabstands-Modus (engl. Incremental Arc Distance Mode)

Der Bogenmittelpunktsabstand ist ein relativer Abstand von der Startposition des Bogens. Standardmäßig ist der Modus Inkrementelle Bogenentfernung (engl. incremental arc distance) eingestellt.

Für einen Bogen, der weniger als 360 Grad beträgt, müssen ein oder mehrere Achsenwörter und ein oder mehrere Offsets programmiert werden.

Für Vollkreise müssen keine Achsenwörter und ein oder mehrere Offsets programmiert werden. Das *P* -Wort ist standardmäßig auf 1 eingestellt und ist optional.

Weitere Informationen zum Modus "Inkrementeller Bogenabstand" finden Sie im Abschnitt G91.1.

Absoluter Bogenabstands-Modus

Bogenmittelpunktsverschiebungen sind der absolute Abstand von der aktuellen 0-Position der Achse.

Für Bögen unter 360 Grad müssen ein oder mehrere Achsenwörter und *beide* Offsets programmiert werden.

Für Vollkreise müssen keine Achsenwörter und ein oder mehrere Offsets programmiert werden. Das *P* -Wort ist standardmäßig auf 1 eingestellt und ist optional.

Weitere Informationen zum Modus *Absoluter Bogenabstand* finden Sie im Abschnitt G90.1.

XY-Ebene (G17)

```
G2 or G3 <X- Y- Z- I- J- P->
```

- *Z* - Helix
- *I* - X offset
- *J* - Y offset
- *P* - Anzahl der Umdrehungen

XZ-Ebene (G18)

```
G2 or G3 <X- Z- Y- I- K- P->
```

- *Y* - Helix
- *I* - X offset
- *K* - Z-Offset
- *P* - Anzahl der Umdrehungen

YZ-Ebene (G19)

```
G2 or G3 <Y- Z- X- J- K- P->
```

- *X* - Helix
- *J* - Y offset

- *K* - Z-Offset
- *P* - Anzahl der Umdrehungen

Es ist ein Fehler, wenn:

- Mit dem Wort F wird keine Vorschubgeschwindigkeit eingestellt.
- Es werden keine Offsets programmiert.
- Wenn der Bogen auf die ausgewählte Ebene projiziert wird, weicht der Abstand zwischen dem aktuellen Punkt und dem Mittelpunkt um mehr als (.05 inch/.5 mm) ODER ((.0005 inch/.005mm) AND .1% des Radius) von dem Abstand zwischen dem Endpunkt und dem Mittelpunkt ab.

Entschlüsselung der Fehlermeldung *Radius am Ende des Bogens unterscheidet sich vom Radius am Anfang:*

- *start* - die aktuelle Position
- *center* - die Mittelposition, wie sie unter Verwendung der Wörter i, j oder k berechnet wird
- *end* - der programmierte Endpunkt
- *r1* - Radius von der Startposition zum Zentrum
- *r2* - Radius von der Endposition zur Mitte

Beispiele für Center-Formate

Das Berechnen von Bögen von Hand kann manchmal schwierig sein. Eine Möglichkeit besteht darin, den Bogen mit einem CAD-Programm zu zeichnen, um die Koordinaten und Offsets zu erhalten. Denken Sie an die oben erwähnte Toleranz, Sie müssen möglicherweise die Präzision Ihres CAD-Programms ändern, um die gewünschten Ergebnisse zu erzielen. Eine weitere Möglichkeit besteht darin, die Koordinaten und den Versatz mithilfe von Formeln zu berechnen. Wie Sie in den folgenden Abbildungen sehen können, kann aus der aktuellen Position, der Endposition und dem Bogenmittelpunkt ein Dreieck gebildet werden.

In der folgenden Abbildung sehen Sie, dass die Startposition X0 Y0 und die Endposition X1 Y1 ist. Der Mittelpunkt des Bogens befindet sich bei X1 Y0. Damit ergibt sich ein Offset von der Startposition von 1 in der X-Achse und 0 in der Y-Achse. In diesem Fall ist nur ein I-Offset erforderlich.

G2-Beispielzeile

```
G0 X0 Y0
G2 X1 Y1 I1 F10 (Bogen im Uhrzeigersinn in der XY-Ebene)
```

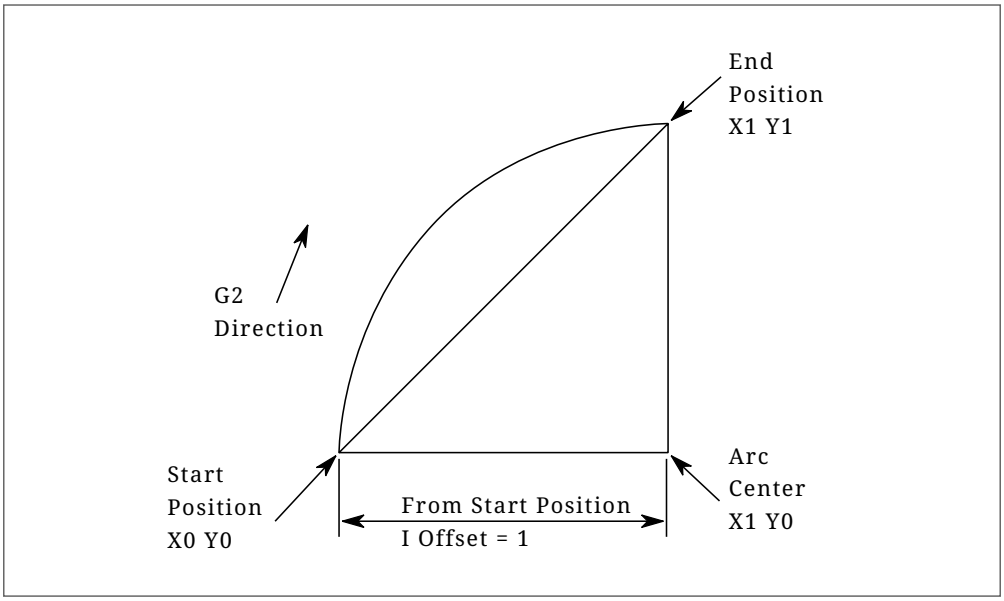


Figure 210. G2-Beispiel

Im nächsten Beispiel sehen wir den Unterschied zwischen den Offsets für Y, wenn wir eine G2 oder eine G3 Bewegung ausführen. Bei der G2-Bewegung ist die Startposition X0 Y0, bei der G3-Bewegung ist sie X0 Y1. Der Mittelpunkt des Bogens liegt für beide Bewegungen bei X1 Y0,5. Bei der G2-Bewegung beträgt der J-Versatz 0,5 und bei der G3-Bewegung -0,5.

G2-G3 Beispielzeile

```
G0 X0 Y0
G2 X0 Y1 I1 J0.5 F25 (Bogen im Uhrzeigersinn in der XY-Ebene)
G3 X0 Y0 I1 J-0.5 F25 (Bogen im Gegenuhrzeigersinn in der XY-Ebene)
```

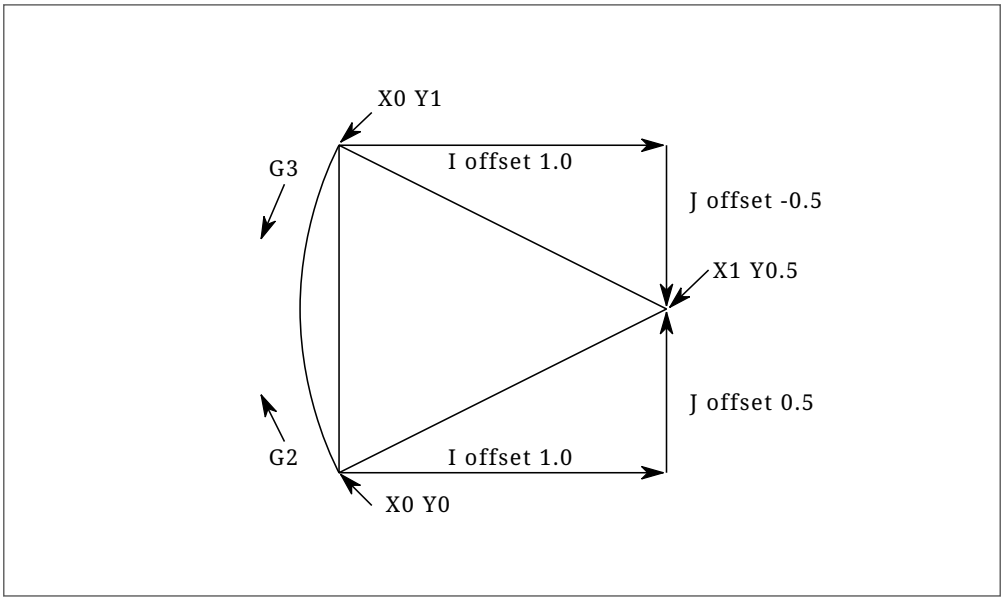


Figure 211. G2-G3 Beispiel

Im nächsten Beispiel zeigen wir, wie der Bogen eine Helix in der Z-Achse bilden kann, indem wir das Z-Wort hinzufügen.

G2 Beispiel Helix

```
G0 X0 Y0 Z0
G17 G2 X10 Y16 I3 J4 Z-1 (Helixbogen mit Z hinzugefügt)
```

Im nächsten Beispiel zeigen wir, wie man mit dem Wort P mehr als einen Zug machen kann.

P-Wort-Beispiel

```
G0 X0 Y0 Z0
G2 X0 Y1 Z-1 I1 J0.5 P2 F25
```

Im Mittelpunktsformat wird der Radius des Bogens nicht angegeben, aber er kann leicht als Abstand vom Mittelpunkt des Kreises zum aktuellen Punkt oder zum Endpunkt des Bogens ermittelt werden.

Bögen im Radiusformat

```
G2 oder G3 Achsen R- <P->
```

- *R* - Radius von der aktuellen Position

Es ist nicht ratsam, Radiusformatbögen zu programmieren, die fast Vollkreise oder fast Halbkreise sind, da eine kleine Änderung der Position des Endpunkts eine viel größere Änderung der Position des Kreismittelpunkts (und damit der Mitte des Bogens) bewirkt. Der Vergrößerungseffekt ist so groß, dass Rundungsfehler in einer Zahl zu Schnitten führen können, die außerhalb der Toleranzen liegen. So führt beispielsweise eine Verschiebung des Endpunkts eines 180-Grad-Bogens um 1 % zu einer Verschiebung des Punkts um 90 Grad entlang des Bogens um 7 %. Nahezu vollständige Kreise sind noch schlimmer. Bögen anderer Größe (im Bereich von winzig bis 165 Grad oder 195 bis 345 Grad) sind in Ordnung.

Im Radiusformat werden die Koordinaten des Endpunktes des Bogens in der ausgewählten Ebene zusammen mit dem Radius des Bogens angegeben. Programm *G2 Achsen R-* (oder verwenden Sie *G3* anstelle von *G2*). R ist der Radius. Die Achsenwörter sind alle optional, außer dass mindestens eines der beiden Wörter für die Achsen in der ausgewählten Ebene verwendet werden muss. Die Zahl R ist der Radius. Ein positiver Radius bedeutet, dass der Bogen um weniger als 180 Grad gedreht wird, während ein negativer Radius eine Drehung von mehr als 180 Grad bedeutet. Wenn der Bogen schraubenförmig ist, wird auch der Wert des Endpunkts des Bogens auf der Koordinatenachse parallel zur Achse der Schraubenlinie angegeben.

Es ist ein Fehler, wenn:

- die beiden Achsenwörter für die Achsen der ausgewählten Ebene werden weggelassen
- der Endpunkt des Bogens ist derselbe wie der aktuelle Punkt.

G2-Beispielzeile

```
G17 G2 X10 Y15 R20 Z5 (Radiusformat mit Bogen)
```

Das obige Beispiel ergibt einen (von der positiven Z-Achse aus gesehen) im Uhrzeigersinn verlaufenden Kreis- oder Helix-/Spiralbogen, dessen Achse parallel zur Z-Achse verläuft und an den Stellen X=10, Y=15

und Z=5 endet, mit einem Radius von 20. Wenn der Startwert von Z gleich 5 ist, handelt es sich um einen Kreisbogen parallel zur XY-Ebene; andernfalls handelt es sich um einen Helixbogen.

11.5.6.G4 Verweilen (engl. dwell)

```
G4 P-
```

- *P* - Sekunden zum Verweilen (Gleitkomma)

Die P-Zahl ist die Zeit in Sekunden, die alle Achsen unbewegt bleiben. Die P-Zahl ist eine Fließkommazahl, so dass auch Sekundenbruchteile verwendet werden können. G4 hat keinen Einfluss auf Spindel, Kühlmittel und E/A.

G4 Beispielzeile

```
G4 P0.5 (wartet 0,5 Sekunden bevor Bewegungen fortfahren)
```

Es ist ein Fehler, wenn:

- die P-Nummer ist negativ oder nicht angegeben.

11.5.7.G5 Kubischer Spline

```
G5 X- Y- <I- J-> P- Q-
```

- *I* - X inkrementeller Offset vom Startpunkt zum ersten Kontrollpunkt
- *J* - Y inkrementeller Versatz vom Startpunkt zum ersten Kontrollpunkt
- *P* - X inkrementeller Offset vom Endpunkt zum zweiten Kontrollpunkt
- *Q* - Y inkrementeller Offset vom Endpunkt zum zweiten Kontrollpunkt

G5 erstellt einen kubischen B-Spline in der XY-Ebene nur mit den Achsen X und Y. P und Q müssen für jeden G5-Befehl angegeben werden.

Für den ersten G5-Befehl in einer Reihe von G5-Befehlen müssen sowohl I als auch J angegeben werden. Für nachfolgende G5-Befehle müssen entweder beide I und J angegeben werden oder keiner von beiden. Wenn I und J nicht angegeben sind, entspricht die Anfangsrichtung dieses Kubiks automatisch der Endrichtung des vorherigen Kubiks (als ob I und J die Negation der vorherigen P und Q wären).

Zum Beispiel, um eine geschwungene N-Form zu programmieren:

G5 Beispiel für einen kubischen Spline

```
G90 G17
G0 X0 Y0
G5 I0 J3 P0 Q-3 X1 Y1
```

Ein zweites gekrümmtes N, das sich nahtlos an dieses anschließt, kann nun ohne Angabe von I und J erstellt werden:

G5 Beispiel eines nachfolgenden kubischen Splines

```
G5 P0 Q-3 X2 Y2
```

Es ist ein Fehler, wenn:

- P und Q sind nicht beide angegeben.
- Es wird nur eines von I oder J angegeben.
- I oder J sind im ersten einer Reihe von G5-Befehlen nicht angegeben.
- Eine andere Achse als X oder Y wird angegeben.
- Die aktive Ebene ist nicht G17.

11.5.8.G5.1 Quadratischer Spline

```
G5.1 X- Y- I- J-
```

- I" - X inkrementeller Offset vom Startpunkt zum Kontrollpunkt
- *J* - Inkrementeller Y-Offset vom Startpunkt zum Kontrollpunkt

G5.1 erzeugt einen quadratischen B-Spline in der XY-Ebene nur mit der X- und Y-Achse. Wenn I oder J nicht angegeben werden, ergibt sich für die nicht angegebene Achse ein Null-Offset, so dass eine oder beide angegeben werden müssen.

Um zum Beispiel eine Parabel durch den Ursprung von X-2 Y4 nach X2 Y4 zu programmieren:

G5.1 Beispiel eines quadratischen Splines

```
G90 G17
G0 X-2 Y4
G5.1 X2 I2 J-8
```

Es ist ein Fehler, wenn:

- sowohl der I- als auch der J-Offset sind nicht spezifiziert oder Null
- Eine andere Achse als X oder Y wird angegeben.
- Die aktive Ebene ist nicht G17.

11.5.9.G5.2 G5.3 NURBS Block

```
G5.2 <P-> <X- Y-> <L->
X- Y- <P->
...
G5.3
```

WARNING G5.2, G5.3 ist experimentell und nicht vollständig getestet.

G5.2 dient zum Öffnen des Datenblocks, der ein NURBS definiert, und G5.3 zum Schließen des Datenblocks. In den Zeilen zwischen diesen beiden Codes werden die Kurvenkontrollpunkte mit ihren zugehörigen "Gewichten" (P) und dem Parameter (L), der die Reihenfolge der Kurve bestimmt, definiert.

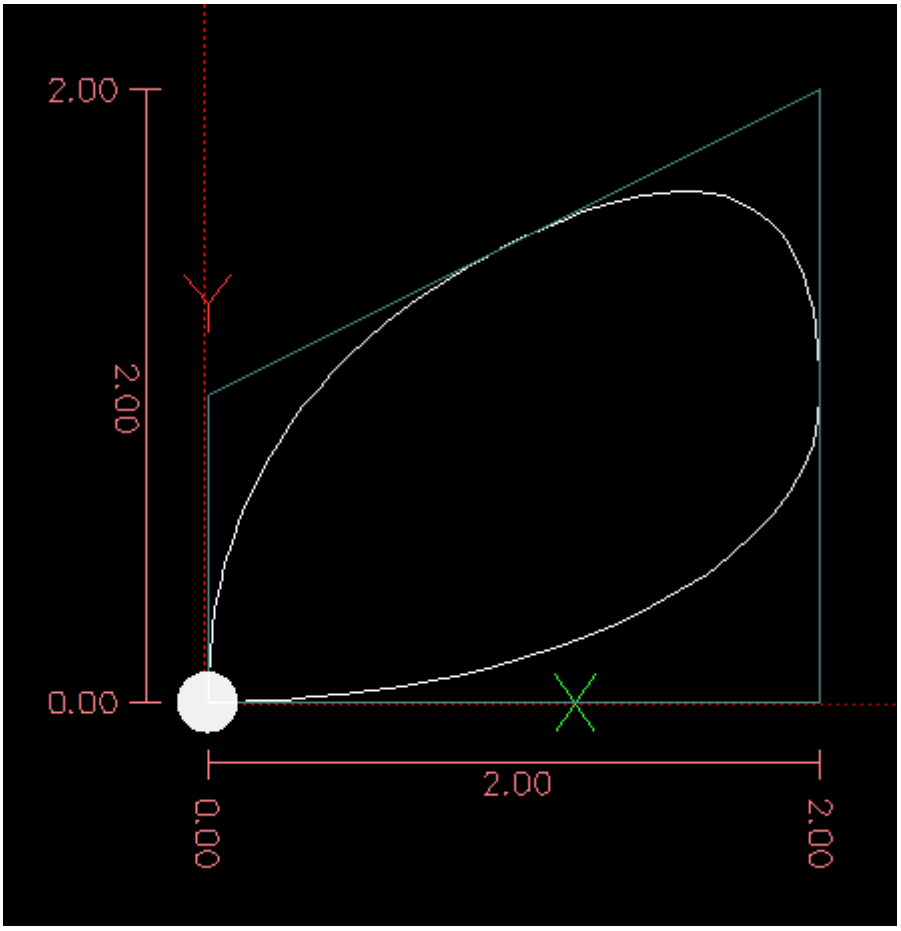
Die aktuelle Koordinate vor dem ersten G5.2-Befehl wird immer als erster NURBS-Kontrollpunkt verwendet. Um das Gewicht für diesen ersten Kontrollpunkt festzulegen, programmieren Sie zunächst G5.2 P- ohne Angabe von X Y.

Ist P nicht angegeben, dann ist das Standardgewicht 1. Ist L nicht angegeben, dann ist die Standardreihenfolge 3.

G5.2 Beispiel

```
G0 X0 Y0 (Eilgang)
F10 (Vorschubgeschwindigkeit einstellen)
G5.2 P1 L3
     X0 Y1 P1
     X2 Y2 P1
     X2 Y0 P1
     X0 Y0 P2
G5.3
; Die schnellen Bewegungen zeigen denselben Weg ohne den NURBS-Block
G0 X0 Y1
   X2 Y2
   X2 Y0
   X0 Y0
M2
```

Beispiel einer NURBS-Ausgabe



Weitere Informationen über NURBS finden Sie hier:

https://wiki.linuxcnc.org/cgi-bin/wiki.pl?NURBS

11.5.10.G7 Drehmaschinen Durchmesser-Modus

```
G7
```

Programmieren Sie G7, um den Durchmessermodus für die Achse X auf einer Drehmaschine aufzurufen. Im Durchmessermodus entspricht die Bewegung der X-Achse auf einer Drehmaschine der Hälfte des Abstands zur Mitte der Drehmaschine. Zum Beispiel würde X1 den Fräser auf 0.500 Zoll von der Mitte der Drehmaschine bewegen, was einen Teil mit 1 Zoll Durchmesser ergibt.

11.5.11.G8 Drehmaschinen Radius-Modus

```
G8
```

Programmieren Sie G8, um den Radiusmodus für die Achse X auf einer Drehmaschine aufzurufen. Im Radiusmodus entspricht die Bewegung der X-Achse auf einer Drehmaschine dem Abstand von der Mitte. Ein Schnitt bei X1 würde also ein Teil mit einem Durchmesser von 2 Zoll ergeben. G8 ist die Standardeinstellung beim Einschalten.

11.5.12.G10 L0 Werkzeug-Tabellendaten neu laden

```
G10 L0
```

G10 L0 Alle Daten der Werkzeugtabelle neu laden. Erfordert, dass kein aktuelles Werkzeug in der Spindel geladen ist.

NOTE Bei Verwendung von G10 L0 werden die Werkzeugparameter (#5401-#5413) sofort aktualisiert und alle geänderten Werkzeugdurchmesser werden für nachfolgende G41,42 Fräserradiuskompensationsbefehle verwendet. Bestehende G43-Werte für die Werkzeuglängenkorrektur bleiben so lange gültig, bis sie durch neue G43-Befehle aktualisiert werden.

11.5.13.G10 L1 Werkzeugtabelle festlegen

```
G10 L1 P- axes <R- I- J- Q->
```

- P" - Werkzeugnummer
- R" - Radius des Werkzeugs
- *I* - vorderer Winkel (Drehmaschine)
- *J* – Rückenwinkel (Drehmaschine)
- *Q* - Ausrichtung (Drehmaschine)

G10 L1 setzt die Werkzeugtabelle für die Werkzeugnummer *P* auf die Werte der Wörter.

Ein gültiges G10 L1 schreibt die Werkzeugtabelle für das angegebene Werkzeug neu und lädt sie neu.

G10 L1 Beispielzeile

```
G10 L1 P1 Z1.5 (Werkzeug 1 Z-Versatz vom Maschinenursprung auf 1.5 setzen)
G10 L1 P2 R0.015 Q3 (Drehbankbeispiel, das den Radius von Werkzeug 2 auf 0.015 und die
Orientierung auf 3 setzt)
```

Es ist ein Fehler, wenn:

- Fräserkompensation ist aktiviert
- Die P-Nummer ist nicht spezifiziert
- Die P-Nummer ist keine gültige Werkzeugnummer aus der Werkzeugtabelle
- Die P-Nummer ist 0

Weitere Informationen über die Werkzeugausrichtung, die durch das *Q*-Wort beschrieben wird, finden Sie im Diagramm Drehmaschinen Werkzeug-Ausrichtung.

11.5.14.G10 L2 Koordinatensystem festlegen

```
G10 L2 P- <Achsen R->
```

- *P* - Koordinatensystem (0-9)
- *R* - Rotation um die Z-Achse

G10 L2 verschiebt den Ursprung der Achsen im angegebenen Koordinatensystem auf den Wert des Achsenwortes. Der Versatz bezieht sich auf den während der Referenzfahrt ermittelten Maschinenursprung. Der Offset-Wert ersetzt alle aktuellen Offsets, die für das angegebene Koordinatensystem gelten. Nicht verwendete Achsenwörter werden nicht geändert.

Programmieren Sie P0 bis P9, um das zu ändernde Koordinatensystem anzugeben.

Table 88. Koordinatensystem

P-Wert	Koordinatensystem	G-Code
0	Aktiv	k.A.
1	1	G54
2	2	G55
3	3	G56
4	4	G57
5	5	G58
6	6	G59
7	7	G59.1
8	8	G59.2
9	9	G59.3

Optional können Sie R programmieren, um die Drehung der XY-Achse um die Z-Achse anzugeben. Die Drehrichtung ist gegen den Uhrzeigersinn, vom positiven Ende der Z-Achse aus gesehen.

Alle Achsenwörter sind optional.

Der inkrementelle Entfernungsmodus (*G91*) hat keine Auswirkungen auf *G10 L2*.

Wichtige Konzepte:

- `G10 L2 Pn` wechselt nicht vom aktuellen Koordinatensystem zu dem durch P angegebenen, Sie müssen mit G54-59.3 ein Koordinatensystem auswählen.
- Wenn eine Drehung in Kraft ist, wird eine Achse nur in positiver oder negativer Richtung bewegt, nicht aber entlang der gedrehten Achse.
- Wenn ein *G52 'lokaler Offset* oder ein *G92*-offset zum Ursprung vor "G10 L2" in Kraft war, bleibt er

auch danach in Kraft.

- Bei der Programmierung eines Koordinatensystems mit R wird jedes *G52* oder *G92* **nach** der Drehung angewendet.
- Das Koordinatensystem, dessen Ursprung durch einen "G10"-Befehl gesetzt wird, kann zum Zeitpunkt der Ausführung des "G10"-Befehls aktiv oder inaktiv sein. Wenn es gerade aktiv ist, werden die neuen Koordinaten sofort wirksam.

Es ist ein Fehler, wenn:

- Die P-Nummer ergibt keine ganze Zahl im Bereich von 0 bis 9.
- Es wird eine Achse programmiert, die nicht in der Konfiguration definiert ist.

G10 L2 Beispielzeile

```
G10 L2 P1 X3.5 Y17.2
```

Im obigen Beispiel wird der Ursprung des ersten Koordinatensystems (das mit *G54* ausgewählte) auf X=3,5 und Y=17,2 gesetzt. Da nur X und Y angegeben sind, wird der Ursprungspunkt nur in X und Y verschoben; die anderen Koordinaten werden nicht verändert.

G10 L2 Beispielzeile

```
G10 L2 P1 X0 Y0 Z0 (Offsets für X-, Y- & Z-Achsen im Koordinatensystem 1 löschen)
```

Im obigen Beispiel werden die XYZ-Koordinaten des Koordinatensystems 1 auf den Maschinenursprung gesetzt.

Das Koordinatensystem wird im Abschnitt Koordinatensystem beschrieben.

11.5.15.G10 L10 Werkzeugtabelle festlegen

```
G10 L10 P- axes <R- I- J- Q->
```

- P" - Werkzeugnummer
- R" - Radius des Werkzeugs
- *I* - vorderer Winkel (Drehmaschine)
- *J* – Rückenwinkel (Drehmaschine)
- *Q* - Ausrichtung (Drehmaschine)

G10 L10 ändert den Eintrag in der Werkzeugtabelle für das Werkzeug P so, dass die aktuellen Koordinaten für die angegebenen Achsen zu den angegebenen Werten werden, wenn die Werkzeugkorrektur neu geladen wird, während sich die Maschine in ihrer aktuellen Position befindet und die aktuellen G5x- und G52/G92-Korrekturen aktiv sind. Die Achsen, die nicht im Befehl G10 L10 angegeben sind, werden nicht geändert. Dies könnte bei einer Messtasterbewegung nützlich sein, wie im Abschnitt G38 beschrieben.

G10 L10 Beispiel

```
T1 M6 G43 (Werkzeug 1 und Werkzeuglängenkorrekturen laden)
G10 L10 P1 Z1.5 (setzt die aktuelle Position für Z auf 1.5)
G43 (Nachladen der Werkzeuglängenkorrekturen aus der geänderten Werkzeugtabelle)
M2 (Programm beenden)
```

- Weitere Informationen finden Sie in den Abschnitten T & M6, und G43/G43.1.

Es ist ein Fehler, wenn:

- Fräserkompensation ist aktiviert
- Die P-Nummer ist nicht spezifiziert
- Die P-Nummer ist keine gültige Werkzeugnummer aus der Werkzeugtabelle
- Die P-Nummer ist 0

11.5.16.G10 L11 Werkzeugtabelle festlegen

```
G10 L11 P- axes <R- I- J- Q->
```

- P" - Werkzeugnummer
- R" - Radius des Werkzeugs
- *I* - vorderer Winkel (Drehmaschine)
- *J* – Rückenwinkel (Drehmaschine)
- *Q* - Ausrichtung (Drehmaschine)

G10 L11 entspricht G10 L10 mit dem Unterschied, dass der Eintrag nicht nach den aktuellen Versätzen eingestellt wird, sondern so, dass die aktuellen Koordinaten zum angegebenen Wert werden, wenn der neue Werkzeugversatz neu geladen wird und die Maschine im G59.3-Koordinatensystem ohne aktiven G52/G92-Versatz platziert wird.

Dadurch kann der Benutzer das G59.3-Koordinatensystem auf einen festen Punkt auf der Maschine einstellen und diese Vorrichtung dann zum Messen von Werkzeugen ohne Rücksicht auf andere derzeit aktive Versätze verwenden.

Es ist ein Fehler, wenn:

- Fräserkompensation ist aktiviert
- Die P-Nummer ist nicht spezifiziert
- Die P-Nummer ist keine gültige Werkzeugnummer aus der Werkzeugtabelle
- Die P-Nummer ist 0

11.5.17.G10 L20 Koordinatensystem festlegen

```
G10 L20 P- axes
```

- *P* - Koordinatensystem (0-9)

G10 L20 ist ähnlich wie G10 L2, mit dem Unterschied, dass der Offset/Eintrag nicht auf den angegebenen Wert gesetzt wird, sondern auf einen berechneten Wert, der die aktuellen Koordinaten zu dem angegebenen Wert macht.

G10 L20 Beispiel Zeile

```
G10 L20 P1 X1.5 (setzt die aktuelle Position der X-Achse im Koordinatensystem 1 auf 1,5)
```

Es ist ein Fehler, wenn:

- Die P-Nummer ergibt keine ganze Zahl im Bereich von 0 bis 9.
- Es wird eine Achse programmiert, die nicht in der Konfiguration definiert ist.

11.5.18.G12.1, G13.1 Select Kinematics

```
G12.1 P-
G13.1
```

G12.1 selects one of the kinematics provided by a switchable kinematics module; the *P* word is the kinematics number. *G13.1* cancels back to identity kinematics, where joints and coordinates are the same thing. Which number that is is declared by the module itself, so *G13.1* means the same kinematics whatever order a module lists its types in.

WARNING Deprecation notice: selecting the kinematics by writing the `motion.switchkins-type` HAL pin is deprecated. It takes the same numbering and still works, but it does not tell the interpreter that anything changed, so a program is read, its limits checked and its path looked ahead in whatever kinematics the interpreter last knew about, which need not be the one that runs it. Use *G12.1* and *G13.1*.

Both codes are queue synchronisation points. The interpreter waits for queued motion to finish before the kinematics changes, so no move is ever planned in one kinematics and executed in another. Because of that, both codes stop any blending that was in progress, in the same way *G4* does.

The kinematics module decides what each number means, and declares which one is identity. See the `switchkins` section of the kins(9) man page for the modules that support switching and the order in which they list their kinematics. A machine whose kinematics module is not switchable rejects the change, and *G13.1* is an error on a switchable module that declares no identity kinematics; such a module can still be driven by number with *G12.1*.

Selecting a kinematics does not move the machine. It changes how joint positions and coordinate positions map onto each other, so the position readout can change even though nothing has moved.

The active kinematics is available to the program as the read-only parameter #<*_kins_type*>, which lets a subroutine put back whatever was selected before it ran:

```
#<saved> = #<_kins_type>
G12.1 P2      (work in kinematics 2)
( ... )
G12.1 P#<saved> (put back whatever the caller was using)
```

Nothing cancels the selection on its own. It survives the end of the program and an abort, so that the kinematics keeps matching what the position readout shows. End a program with *G13.1* if it should leave the machine in identity kinematics.

G12.1, G13.1 Example

```
G12.1 P1  (switch to kinematics 1)
G0 X0 Y0
G13.1     (back to identity kinematics)
```

Es ist ein Fehler, wenn:

- *G12.1* is used without a *P* word.
- The *P* word is negative.
- The *P* word names a kinematics the module does not provide.
- A *P* word is used with *G13.1*.
- *G13.1* is used and the module declares no identity kinematics.

11.5.19.G17 - G19.1 Ebene auswählen

Diese Codes stellen die aktuelle Ebene wie folgt ein:

- *G17* - XY (Standard)
- *G18* - ZX
- *G19* - YZ
- *G17.1* - UV
- *G18.1* - WU
- *G19.1* - VW

Die UV-, WU- und VW-Ebenen unterstützen keine Bögen.

Es ist eine gute Idee, eine Ebenenauswahl in die Präambel jeder G-Code-Datei aufzunehmen.

Die Auswirkungen der Auswahl einer Ebene werden in den Abschnitten G2 G3 Bögen und G81 G89 behandelt.

11.5.20.G20, G21 Einheiten

- *G20* - um Zoll als Längeneinheit zu verwenden.
- *G21* - um Millimeter als Längeneinheiten zu verwenden.

Es ist eine gute Idee, Einheiten in die Präambel jeder G-Code-Datei aufzunehmen.

11.5.21.G28, G28.1 Gehe zu/Setze vordefinierte Position

WARNING	Verwenden Sie G28 nur, wenn Ihre Maschine auf eine wiederholbare Position referenziert ist und die gewünschte G28-Position mit G28.1 gespeichert wurde.

G28 verwendet die in Parameter 5161-5169 gespeicherten Werte als X Y Z A B C U V W Endpunkt zum Anfahren. Die Parameterwerte sind *absolute* Maschinenkoordinaten in den maschineneigenen *Einheiten* wie in der INI-Datei angegeben. Alle in der INI-Datei definierten Achsen werden bewegt, wenn ein G28 ausgegeben wird. Wenn keine Positionen mit G28.1 gespeichert werden, dann werden alle Achsen auf den Maschinenursprung gehen.

- *G28* - bewegt sich im Schnellauf durch von der aktuellen Position zur *absoluten* Position der Werte in den Parametern 5161-5166.
- *G28 Achsen* - bewegt sich im Schnellauf zu der durch *Achsen* angegebenen Position, einschließlich aller Offsets, und dann weiter im Schnellauf zu der *absoluten* Position der Werte in den Parametern 5161-5166 für solche *Achsen*, für die der G28-Aufruf Positionsparameter erhalten hat. Jede *Achse*, die durch G28 nicht beschrieben ist, wird nicht bewegt.
- *G28.1* - speichert die aktuelle *absolute* Position in den Parametern 5161-5166.

G28 Beispielzeile

```
G28 Z2.5 (schnell auf Z2.5 und dann auf die in #5163 angegebene Z-Position)
```

Es ist ein Fehler, wenn :

- Fräserausgleich (engl. cutter compensation) aktiviert ist

11.5.22.G28.2 Home from G-code

This non-modal code lets a program or MDI line reference the machine instead of requiring the operator to use the GUI's **Home All** button. It follows the same modal-group-0 pattern as `G28.1`/`G30.1`, takes no axis words, and **requires** a `P` word saying what to home.

- *G28.2 P-1* - runs the homing cycle on every joint, in `HOME_SEQUENCE` order (the same operation as the GUI **Home All**). This form requires the config to set `[JOINT_n]HOME_SEQUENCE`; exactly as with **Home All**, a config with no `HOME_SEQUENCE` cannot home all joints at once and the command fails, leaving the machine in joint mode.
- *G28.2 Pn* - runs the homing cycle on joint *n* only, where *n* is the 0-based joint number matching its `[JOINT_n]` INI section (the same numbering used by `HOME_SEQUENCE` and by joint jogging). Other joints are left as they are, and `HOME_SEQUENCE` is not required for the `Pn` form. How many joints actually home depends on the `HOME_SEQUENCE` of joint *n*:
  - a **negative** (synchronized) sequence homes the whole synchronized group - `Pn` on either joint of a gantry pair homes both;
  - a **positive** sequence shared with other joints homes only joint *n* - use `G28.2 P-1` to home the

shared group together;

- no sequence, or a sequence not shared with any other joint, homes just joint *n*.

WARNING There is no bare G28.2. Homing drives joints onto their switches at homing speed, ignoring soft limits, from wherever the machine happens to be, so homing the whole machine has to be asked for explicitly with P-1 rather than being what a truncated or mistyped line does. A G28.2 with no P word is an error and stops the program.

G28.2 Example Lines

```
G28.2 P-1   (home all joints, in HOME_SEQUENCE order)
G28.2 P1    (home joint 1 only)
```

A queued G28.2 dips motion into free mode for the duration of the homing cycle and restores whatever mode (manual/MDI/auto) was active once it finishes, so the mode dip is invisible at the task level. Motion still enforces its own safety: the home is honored only when the machine is idle (in position with no queued motion) or in joint mode, and a home is refused mid-motion. Homing inhibits and per-joint limit handling are unchanged.

NOTE When the homing cycle finishes, the interpreter resyncs its model of the current position from the machine (the same way it does after probing or a tool change), so a following move made in incremental distance mode (G91), or an arc whose center is given with I/J/K, is computed from where the machine actually is after homing. G28.2 is a queue point: read ahead stops at it and resumes once the cycle has run. G5x/G92 offsets that were active before the G28.2 are still applied afterward and now refer to the newly established machine zero; clear or re-establish them if that is not what you want.

NOTE There is no G-code unhome. Clearing a joint's reference is done from the GUI, halui or linuxcncrsh.

NOTE G28.2 is a LinuxCNC extension; there is no standard Fanuc equivalent.

Es ist ein Fehler, wenn:

- Cutter radius compensation is active (G41, G41.1, G42 or G42.1).
- An axis word is present (*G28.2* takes no axis words).
- The *P* word is missing.
- The *P* word is not a whole number, or is negative other than P-1.
- The *P* word names a joint number the machine does not have.

11.5.23.G30, G30.1 Gehe zu/Setze vordefinierte Position

WARNING Verwenden Sie G30 nur, wenn Ihre Maschine auf eine wiederholbare Position ausgerichtet ist und die gewünschte G30-Position mit G30.1 gespeichert wurde.

G30 funktioniert wie G28, verwendet aber die in Parameter 5181-5189 gespeicherten Werte als X Y Z A B C U V W Endpunkt zum Anfahren. Die Parameterwerte sind *absolute* Maschinenkoordinaten in den maschineneigenen *Einheiten* wie in der INI-Datei angegeben. Alle in der INI-Datei definierten Achsen werden bewegt, wenn ein G30 ausgegeben wird. Wenn keine Positionen mit G30.1 gespeichert werden, dann gehen alle Achsen zum Maschinenursprung.

NOTE	G30-Parameter werden verwendet, um das Werkzeug zu bewegen, wenn ein M6 programmiert wird, sofern TOOL_CHANGE_AT_G30=1 in der [EMCIO]-Sektion der INI-Datei steht.

- *G30* - führt einen Eilgang von der aktuellen Position zur *absoluten* Position der Werte in den Parametern 5181-5189 durch.
- *G30 Achsen* - führt eine Eilbewegung zu der durch *Achsen* angegebenen Position aus, einschließlich aller Offsets, und führt dann eine Eilbewegung zu der *absoluten* Position der Werte in den Parametern 5181-5189 für alle *Achsen* angegebenen aus. Jede *Achse*, die nicht angegeben ist, wird nicht bewegt.
- *G30.1* - speichert die aktuelle absolute Position in den Parametern 5181-5186.

G30 Beispielzeile

```
G30 Z2.5 (schnell zu Z2.5 und dann zu der in #5183 angegebenen Z-Position)
```

Es ist ein Fehler, wenn :

- Fräserausgleich (engl. cutter compensation) aktiviert ist

11.5.24.G33 Spindelsynchronisierte Bewegung

```
G33 X- Y- Z- K- $- D-
```

- *K* - Weg pro Umdrehung
- *D* - (optional) start angle offset from the spindle index pulse, in degrees

Für eine spindelsynchrone Bewegung in einer Richtung codieren Sie *G33 X- Y- Z- K-*, wobei K die in XYZ zurückgelegte Strecke pro Spindelumdrehung angibt. Wenn Sie zum Beispiel bei *Z=0* beginnen, erzeugt *G33 Z-1 K.0625* eine Bewegung von 1 Zoll in Z über 16 Umdrehungen der Spindel. Dieser Befehl könnte Teil eines Programms zur Herstellung eines 16TPI-Gewindes sein. Ein weiteres Beispiel im metrischen System: *G33 Z-15 K1.5* erzeugt eine Bewegung von 15 mm, während sich die Spindel 10 Mal dreht, um ein Gewinde von 1,5 mm herzustellen.

Das (optionale) Argument $ legt fest, mit welcher Spindel die Bewegung synchronisiert wird (Standard ist Null). Zum Beispiel bewegt G33 Z10 K1 $1 die Spindel synchron mit dem Wert des Pins spindle.N.revs HAL.

Die spindelsynchronisierte Bewegung wartet auf den Spindelindex und die Spindel an den Drehzahlstiften, so dass mehrere Durchgänge aneinandergereiht werden. *G33* bewegt das Ende am programmierten Endpunkt. G33 kann zum Schneiden von kegelförmigen Gewinden oder einem Konus

verwendet werden.

The (optional) *D* argument delays the start of the motion until the spindle has turned *D* degrees past the index pulse, instead of starting at the index pulse itself. If *D* is omitted the default is zero, which starts at the index pulse as before. This is how successive starts of a multi-start thread are cut: the same thread program is run once per start, each with a different *D*. For example, a two-start thread is cut with *D0* and then *D180*; a three-start thread with *D0*, *D120* and *D240*.

NOTE *D* is an angle past the index pulse and so is only meaningful as a positive number. A negative value is not an error: its magnitude is used, so *D-90* behaves exactly as *D90*. Values of 360 or more are not reduced, so *D450* waits a full turn longer than *D90*.

Alle Achsenwörter sind optional, außer dass mindestens eines verwendet werden muss.

NOTE K folgt der durch "X- Y- Z-" beschriebenen Antriebslinie. K ist nicht parallel zur Z-Achse, wenn X- oder Y-Endpunkte verwendet werden, z. B. beim Schneiden von kegelförmigen Gewinden.

NOTE The spindle speed override has no effect during a G33 move, and the previous setting is restored when the move ends. The commanded position follows the spindle, but the axis lags it by an amount proportional to speed, so changing the speed part way through shifts the tool along the helix it is already cutting. A G76 cycle holds the override at the value the cycle started with rather than dropping it, because the whole cycle is one block; a G33 thread is one pass per line with the retract in between, so there is no span to hold.

Technische Informationen

A G33 move waits for the spindle index, then accelerates from rest to the synchronized feed. The commanded position follows the spindle angle measured from that index, so a given thread is cut in the same place whatever the spindle speed.

While the axis is still accelerating it falls behind that position, by roughly the pitch squared times the spindle speed squared over twice the axis acceleration limit, and it makes the distance up over the rest of the move. Allow enough lead-in for it to catch up before the cut begins, or the start of the thread is cut with the wrong lead. LinuxCNC reports *lead-in too short to reach sync* when the move ends before the axis has caught up. A small lag proportional to speed remains for the rest of the move, which is why the spindle speed must not change while a thread is being cut.

When a *D* offset is programmed the axis is instead held at rest until the spindle reaches the requested angle, and tracking begins from there. Allow enough clearance before the thread for the axis to accelerate from that point.

HAL-Verbindungen

Der Pin *spindle.N.at-speed* muss gesetzt oder auf true gesetzt werden, damit die Bewegung beginnt. Außerdem muss *spindle.N.revs* bei jeder Umdrehung der Spindel um 1 erhöht werden und der Pin *spindle.N.index-enable* muss mit einem Encoder- (oder Resolver-) Zähler verbunden sein, der *index-enable* einmal pro Umdrehung zurücksetzt.

Weitere Informationen zur spindelsynchronisierten Bewegung finden Sie im Integrators Manual.

G33 Beispiel

```
G90 (absoluter Abstandsmodus)
G0 X1 Z0.1 (Eilgang auf Position)
S100 M3 (Spindeldrehung starten)
G33 Z-2 K0.125 (Z-Achse mit einer Geschwindigkeit von 0,125 pro Umdrehung auf -2 fahren)
G0 X1.25 (Werkzeug im Eilgang vom Werkstück wegfahren)
Z0.1 (Eilgang auf Z-Startposition)
M2 (Programm beenden)
```

- Siehe G90 & G0 & M2 für weitere Informationen.

Es ist ein Fehler, wenn:

- Alle Achsenwörter werden weggelassen.
- No K word is given.
- An F word is given (the feed follows from K and the spindle speed).
- The selected spindle is not commanded to turn (M3 or M4 active) when this command is executed.
- The pitch and the spindle speed together ask an axis to move faster than its maximum velocity.

NOTE In constant surface speed mode the spindle speed follows the radius, so the check uses the fastest speed the move can reach: the speed at its smallest cutting radius, limited by the G96 *D* word and by `[SPINDLE_n]MAX_FORWARD_VELOCITY`. If the move reaches the centre of rotation and neither limit is set the speed is unbounded and no check is made.

11.5.25.G33.1 Starres Gewindeschneiden

```
G33.1 X- Y- Z- K- I- $-
```

- *K* - Weg pro Umdrehung
- *I* - optional spindle speed multiplier for faster return move (values less than 1 are treated as 1)
- *$* - optionaler Spindelselektor

WARNING Für reines Z-Gewindeschneiden positionieren Sie die XY-Position vor dem Aufruf von G33.1 und verwenden nur ein Z-Wort in G33.1. Wenn die angegebenen Koordinaten nicht die aktuellen Koordinaten beim Aufruf von G33.1 für das Gewindeschneiden sind, erfolgt die Bewegung nicht entlang der Z-Achse, sondern ist eine koordinierte, spindelsynchrone Bewegung von der aktuellen Position zur angegebenen Position und zurück.

Für starres Gewindebohren (Spindel synchronisierte Bewegung mit Rückkehr), Code *G33.1 X- Y- Z- K-*, wobei *K-* die Strecke angibt, die für jede Umdrehung der Spindel zurückgelegt wird.

Eine starres Gewindeschneiden besteht aus der folgenden Sequenz:

- Eine Bewegung von der aktuellen Koordinate zur angegebenen Koordinate, synchronisiert mit der gewählten Spindel im angegebenen Verhältnis und ausgehend von der aktuellen Koordinate mit einem Spindelindeximpuls.
- Bei Erreichen des Endpunkts ein Befehl zur Umkehrung der Spindel und zur Erhöhung der Geschwindigkeit um einen durch den Multiplikator festgelegten Faktor (z. B. von Rechts- auf Linkslauf).
- Fortgesetzte synchronisierte Bewegung über die angegebene Endkoordinate hinaus, bis die Spindel tatsächlich anhält und reversiert.
- Fortsetzung der synchronisierten Bewegung zurück zur ursprünglichen Koordinate.
- Bei Erreichen der Originalkoordinate wird der Befehl gegeben, die Spindel ein zweites Mal umzudrehen (z. B. von Links- auf Rechtslauf).
- Fortgesetzte synchronisierte Bewegung über die ursprüngliche Koordinate hinaus, bis die Spindel tatsächlich anhält und reversiert.
- Eine **unsynchronisierte** Bewegung zurück zur ursprünglichen Koordinate.

Spindelsynchronisierte Bewegungen warten auf den Spindelindex, so dass mehrere Durchgänge aneinandergereiht werden.*G33.1*-Bewegungen enden an der ursprünglichen Koordinate.

Alle Achsenwörter sind optional, außer dass mindestens eines verwendet werden muss.

G33.1 Beispiel

```
G90 (Absolutmodus einstellen)
G0 X1.000 Y1.000 Z0.100 (Eilgang auf Startposition)
S100 M3 (Spindel einschalten, 100 RPM)
G33.1 Z-0.750 K0.05 (Gewindebohrer 20 TPI, 0,750 tief)
M2 (Endprogramm)
```

- Siehe G90 & G0 & M2 für weitere Informationen.

NOTE Unlike G33, the spindle speed override stays active during a G33.1 move and the feed follows whatever the spindle actually does. A tap guides itself in its own hole, so there is no thread lead to spoil, and being able to slow the spindle while the tap is in the work is useful.

Es ist ein Fehler, wenn:

- Alle Achsenwörter werden weggelassen.
- No K word is given.
- An F word is given (the feed follows from K and the spindle speed).
- The selected spindle is not commanded to turn (M3 or M4 active) when this command is executed.
- The pitch and the spindle speed together ask an axis to move faster than its maximum velocity. See G33 for how the speed is determined in constant surface speed mode.

11.5.26.G38.*n* Gerade Sonde (engl. straight probe)

```
G38.n Achsen
```

- G38.2" - Messtaster zum Werkstück, Stopp bei Berührung, Signalfehler bei Ausfall
- *G38.3* - Sonde in Richtung Werkstück, Stopp bei Kontakt
- *G38.4* - Messtaster vom Werkstück weg, Stopp bei Kontaktverlust, Fehlersignal bei Ausfall
- *G38.5* - Sonde weg vom Werkstück, Stopp bei Kontaktverlust

IMPORTANT

Sie können eine Tasterbewegung erst dann verwenden, wenn Ihre Maschine so eingerichtet ist, dass sie ein Taster-Eingangssignal liefert. Das Taster-Eingangssignal muss mit *motion.probe-input* in einer .hal-Datei verbunden sein. G38.*n* verwendet motion.probe-input, um festzustellen, wann der Messtaster den Kontakt hergestellt (oder verloren) hat. TRUE für einen geschlossenen (sich berührenden) Tasterkontakt, FALSE für einen offenen Tasterkontakt.

Programmieren Sie *G38.n Achsen*, um einen geraden Messtasterbetrieb durchzuführen. Die Achsenwörter sind optional, außer dass mindestens eines von ihnen verwendet werden muss. Die Achsenwörter definieren zusammen den Zielpunkt, den der Messtaster ausgehend von der aktuellen Position anfahren wird. Wird der Messtaster nicht ausgelöst, bevor der Zielpunkt erreicht ist, melden G38.2 und G38.4 einen Fehler.

Das Werkzeug in der Spindel muss ein Taster sein oder einen Tasterschalter berühren.

Als Reaktion auf diesen Befehl bewegt die Maschine den kontrollierten Punkt (der sich in der Mitte der Tastkugel befinden sollte) in einer geraden Linie mit dem aktuellen Vorschub zum programmierten Punkt. Im Modus Inverser Zeitvorschub (engl. inverse time feed mode) ist die Vorschubgeschwindigkeit so gewählt, dass die gesamte Bewegung vom aktuellen Punkt zum programmierten Punkt die angegebene Zeit dauern würde. Die Bewegung stoppt (innerhalb der Maschinenbeschleunigungsgrenzen), wenn der programmierte Punkt erreicht ist oder wenn die geforderte Änderung des Tastereingangs erfolgt, je nachdem, was zuerst eintritt.

Table 89. Sondieren mit G-Codes

Code	Zielzustand	Orientierung der Bewegung	Fehlersignal
G38.2	Berührt (engl. touched)	Zum Stück hin	Ja
G38.3	Berührt (engl. touched)	Zum Stück hin	Nein
G38.4	Unberührt (engl. untouched)	Vom Stück weg	Ja
G38.5	Unberührt (engl. untouched)	Vom Stück weg	Nein

Nach erfolgreicher Antastung werden die Parameter #5061 bis #5069 auf die X-, Y-, Z-, A-, B-, C-, U-, V-, W-Koordinaten der Position des kontrollierten Punktes zum Zeitpunkt der Zustandsänderung des Messtasters (im aktuellen Arbeitskoordinatensystem) gesetzt. Nach erfolgloser Antastung werden sie auf die Koordinaten des programmierten Punktes gesetzt. Der Parameter 5070 wird auf 1 gesetzt, wenn die Antastung erfolgreich war, und auf 0, wenn die Antastung fehlgeschlagen ist. Wenn der Antastvorgang fehlgeschlagen ist, signalisieren G38.2 und G38.4 einen Fehler, indem sie eine Meldung auf dem Bildschirm ausgeben, sofern die gewählte GUI dies unterstützt. Und durch Anhalten der Programmausführung.

Hier ist ein Beispiel für eine Formel zur Sondierung der Werkzeughöhe mit Umwandlung von einem lokalen Koordinatensystem Z versetzt zu in dem Werkzeugtisch gespeicherten Maschinenkoordinaten. Die bestehende Werkzeughöhenkompensation wird zunächst mit G49 aufgehoben, um die Aufnahme in die Berechnung der Höhe zu vermeiden, und die neue Höhe wird aus der Werkzeug-Tabelle geladen. Die Startposition muss hoch genug über der Werkzeughöhensonde sein, um die Verwendung von G49 zu kompensieren.

G38.2 Beispiel

```
G49
G38.2 Z-100 F100
#<zworkoffset> = [#[5203 + #5220 * 20] + #5213 * #5210]
G10 L1 P#5400 Z#<zworkoffset> (set new tool offset)
G43
```

Probe results in #5061 to #5069 are expressed in the current work coordinate system, not in machine coordinates. To convert one, add the offset between the two frames. That offset is the difference between the current position reported in both frames at once: #5021 to #5029 hold the absolute machine position and #5420 to #5428 hold the work position. Because the offset is the same everywhere, it can be read at any later position and still applies to the earlier probe result.

Reading the machine coordinates of a probe

```
G38.2 Z-100 F100
#<zmachine> = [#5063 + #5023 - #5422]
```

#5063 is the probed Z in the work frame, and #5023 - #5422 is the machine-to-work Z offset (current machine Z minus current work Z). The same pattern applies to the other axes using their respective parameters. This holds whichever coordinate system, G92/G52 or tool length offset is active, as long as no coordinate-system rotation is in effect.

Ein Kommentar der Form *(PROBEOPEN dateiname.txt)* öffnet *dateiname.txt* und speichert darin die 9-stellige Koordinate, bestehend aus XYZABCUVW, jeder erfolgreichen geraden Sonde. Die Datei muss mit *(PROBECLOSE)* geschlossen werden. Weitere Informationen finden Sie im Abschnitt Kommentare.

Eine Beispieldatei *smartprobe.ngc* ist enthalten (im Verzeichnis *examples*), um zu demonstrieren, wie die Koordinaten eines Werkstücks mit Hilfe von Tasterbewegungen in eine Datei geschrieben werden. Das Programm *smartprobe.ngc* kann mit minimalen Änderungen mit *ngcgui* verwendet werden.

Es ist ein Fehler, wenn:

- der aktuelle Punkt ist derselbe wie der programmierte Punkt.
- es wird kein Achswort verwendet
- Fräserkompensation ist aktiviert
- der Vorschub ist null
- die Sonde befindet sich bereits im Zielzustand

11.5.27.G40 Kompensation aus

- *G40* -schaltet Fräserkompensation aus. Wenn die Werkzeugkompensation eingeschaltet war, muss die nächste Bewegung eine lineare Bewegung und länger als der Werkzeugdurchmesser sein. Es ist in Ordnung, die Kompensation auszuschalten, wenn sie bereits ausgeschaltet ist.

G40 Beispiel

```
; Aktuelle Position ist X1 nach Beendigung der kompensierten Fräserbewegung
G40 (Kompensation ausschalten)
G0 X1.6 (lineare Bewegung länger als der aktuelle Fräserdurchmesser)
M2 (Programm beenden)
```

Siehe die Abschnitte G0 und M2 für weitere Informationen.

Es ist ein Fehler, wenn:

- Ein G2/G3-Bogenzug folgt direkt nach einem G40.
- Die lineare Bewegung nach Ausschalten der Kompensation ist kleiner als der Werkzeugdurchmesser.

11.5.28.G41 G42 Fräserkompensation

```
G41 <D-> (links vom programmierten Pfad)
G42 <D-> (rechts vom programmierten Pfad)
```

- *D* - Werkzeugnummer

Das D-Wort ist optional; ist kein D-Wort vorhanden, wird der Radius des aktuell geladenen Werkzeugs verwendet (ist kein Werkzeug geladen und kein D-Wort angegeben, wird ein Radius von 0 verwendet).

Falls angegeben, ist das D-Wort die zu verwendende Werkzeugnummer. Normalerweise ist dies die Nummer des Werkzeugs in der Spindel (in diesem Fall ist das D-Wort überflüssig und muss nicht angegeben werden), aber es kann jede gültige Werkzeugnummer sein.

NOTE G41/G42 D0' ist ein wenig speziell. Er verhält sich auf Maschinen mit zufälligem Werkzeugwechsler anders als auf Maschinen mit nicht zufälligem Werkzeugwechsler (siehe Abschnitt Tool Change). Auf Maschinen mit nicht-zufälligem Werkzeugwechsler wendet *G41/G42 D0* den Werkzeuglängenversatz (engl. tool length offset, kurz TLO) des Werkzeugs an, das sich gerade in der Spindel befindet, oder eine TLO von 0, wenn sich kein Werkzeug in der Spindel befindet. Auf Maschinen mit wahlfreiem

> Werkzeugwechsel wendet *G41/G42 D0* die TLO des in der Werkzeugtabelle definierten Werkzeugs T0 an (oder verursacht einen Fehler, wenn T0 nicht in der Werkzeugtabelle definiert ist).

Um die Fräskompensation links vom Werkstückprofil zu starten, verwenden Sie G41. G41 startet die Fräskompensation links von der programmierten Linie, vom positiven Ende der Achse aus gesehen, die senkrecht zur Ebene steht.

Um die Fräskompensation rechts vom Werkstückprofil zu starten, verwenden Sie G42. G42 startet die Fräskompensation rechts von der programmierten Linie, vom positiven Ende der Achse aus gesehen, die senkrecht zur Ebene steht.

Die Anfahrbewegung muss mindestens so lang wie der Werkzeugradius sein. Die Anfahrbewegung kann eine Eilfahrt sein.

Bei aktiver XY-Ebene oder XZ-Ebene kann eine Fräserkompensation durchgeführt werden.

Benutzerbefehle M100-M199 sind zulässig, wenn die Fräserkompensation aktiviert ist.

Das Verhalten des Bearbeitungszentrums bei eingeschalteter Schneidradius-Kompensation wird im Abschnitt Schneidradius-Kompensation zusammen mit Codebeispielen beschrieben.

Es ist ein Fehler, wenn:

- Die D-Nummer ist eine ungültige Werkzeugnummer oder 0.
- Die YZ-Ebene ist aktiv.
- Die Fräserkompensation wird angewiesen, sich einzuschalten, wenn sie bereits eingeschaltet ist.

11.5.29.G41.1, G42.1 Dynamische Fräserkompensation

```
G41.1 D- <L-> (links vom programmierten Pfad)
G42.1 D- <L-> (rechts vom programmierten Pfad)
```

- *D* - Fräserdurchmesser
- *L* - Werkzeugausrichtung (siehe Drehmaschinen Werkzeugausrichtung)

G41.1 & G42.1 funktionieren genauso wie G41 & G42 mit der zusätzlichen Möglichkeit, den Werkzeugdurchmesser zu programmieren. Das L-Wort ist standardmäßig 0, wenn es nicht spezifiziert ist.

Es ist ein Fehler, wenn:

- Die YZ-Ebene ist aktiv.
- Die L-Nummer liegt nicht im Bereich von 0 bis einschließlich 9.
- Die L-Nummer wird verwendet, wenn die XZ-Ebene nicht aktiv ist.
- Die Fräserkompensation wird angewiesen, sich einzuschalten, wenn sie bereits eingeschaltet ist.

11.5.30.G43 Werkzeuglängen-Korrektur

```
G43 <H->
```

- *H* - Werkzeugnummer (optional)
- G43' - aktiviert die Werkzeuglängenkompensation. G43 ändert die nachfolgenden Bewegungen, indem die Achsenkoordinaten um die Länge des Versatzes verschoben werden. G43 verursacht keine Bewegung. Wenn eine kompensierte Achse das nächste Mal bewegt wird, ist der Endpunkt dieser Achse die kompensierte Position.

G43 ohne H-Wort verwendet das aktuell geladene Tool aus dem letzten *Tn M6*.

G43 Hn verwendet den Offset für Werkzeug *n*.

The tool length compensation values actually applied to motion are stored in the numbered parameters *5081-5089*. The numbered parameters *5401-5409* instead hold the loaded tool's stored offset, refreshed on tool change (*M6*) and *G10 L1*/*L10*/*L11*.

NOTE

G43 H0 ist ein wenig speziell. Sein Verhalten unterscheidet sich auf Maschinen mit zufälligem Werkzeugwechsler und Maschinen ohne zufälligen Werkzeugwechsler (siehe Abschnitt Werkzeugwechsler). Bei Maschinen mit nicht zufälligem Werkzeugwechsler wendet *G43 H0* den Werkzeuglängenversatz (kurz TLO für engl. tool length offset) des Werkzeugs an, das sich derzeit in der Spindel befindet, oder einen TLO von 0, wenn sich kein Werkzeug in der Spindel befindet. Auf Werkzeugwechselmaschinen wendet *G43 H0* die TLO des Werkzeugs T0 an, die in der Werkzeugtabellendatei definiert ist (oder verursacht einen Fehler, wenn T0 nicht in der Werkzeugtabelle definiert ist).

G43 H- Beispiel Zeile

```
G43 H1 (Werkzeugkorrekturen mit den Werten von Werkzeug 1 in der Werkzeugtabelle
einstellen)
```

Es ist ein Fehler, wenn:

- die H-Nummer ist keine ganze Zahl, oder
- die H-Zahl ist negativ, oder
- die H-Nummer ist keine gültige Werkzeugnummer (beachten Sie jedoch, dass 0 eine gültige Werkzeugnummer auf Maschinen mit nicht-zufälligem Werkzeugwechsler ist, sie bedeutet "das aktuell in der Spindel befindliche Werkzeug")

11.5.31.G43.1 Dynamischer Werkzeuglängenversatz

```
G43.1-Achsen
```

- *G43.1-Achsen* - Ändern Sie nachfolgende Bewegungen, indem Sie den/die aktuellen Offset(s) der Achsen ersetzen. G43.1 verursacht keine Bewegung. Wenn eine kompensierte Achse das nächste Mal bewegt wird, ist der Endpunkt dieser Achse die kompensierte Position.

G43.1 Beispiel

```
G90 (Absolut-Modus einstellen)
T1 M6 G43 (Werkzeug 1 und Werkzeuglängenkorrekturen laden, Z ist auf Maschine 0 und DRO
zeigt Z1.500)
G43.1 Z0.250 (ersetze Werkzeugkorrektur mit 0.250, DRO zeigt jetzt Z0.250 an)
M2 (Programm beenden)
```

- Siehe Abschnitte G90, T und M6 weitere Informationen.

Es ist ein Fehler, wenn:

- Bewegung wird in der gleichen Zeile wie *G43.1* befohlen

NOTE G43.1 schreibt nicht in die Werkzeugtabelle.

11.5.32.G43.2 Zusätzlichen Werkzeuglängenversatz anwenden

```
G43.2 H- oder Achsen-
```

- *H* - Werkzeugnummer
- *G43.2 Hn* - wendet einen zusätzlichen simultanen Werkzeug-Offset nachfolgenden Bewegungen an, indem zu dem Versatz von Werkzeug *n* hinzuaddiert wird.
- *G43.2 Achsen* - wendet einen zusätzlichen simultanen Werkzeug-Offset für nachfolgende Bewegungen an, in dem es zu den Werten von Achsen-Wörten (engl. axis words) hinzuaddiert.

G43.2 Hn Beispiel

```
G90 (Absolutmodus einstellen)
T1 M6 (Werkzeug 1 laden)
G43 (oder G43 H1 - alle Werkzeugkorrekturen durch die Offsets/Korrekturen von T1
ersetzen)
G43.2 H10 (fügen Sie auch die Werkzeugkorrektur von T10 ein)
M2 (Programm beenden)
```

G43.2 Achsen Beispiel

```
G90 (Absolutmodus einstellen)
T1 M6 (Werkzeug 1 laden)
G43 (oder G43 H1 - alle Werkzeugkorrekturen durch die Offsets/Korrekturen von T1
ersetzen)
G43.2 X0.01 Z0.02 (hinzufügen von 0,01 zum Werkzeug-Versatz in Richtung der X-Achse und
0,02 in Z-Richtung)
M2 (Programm beenden)
```

Sie können eine beliebige Anzahl von Offsets zusammenzählen, indem Sie G43.2 mehrmals aufrufen. Es gibt keine eingebauten Annahmen darüber, welche Zahlen Geometrie-Offsets und welche Verschleiß-Offsets sind, oder dass Sie nur eine von beiden haben sollten.

Wie die anderen G43-Befehle führt auch G43.2 zu keiner Bewegung. Wenn eine kompensierte Achse das

nächste Mal bewegt wird, ist der Endpunkt dieser Achse die kompensierte Position.

Es ist ein Fehler, wenn:

- *H* ist nicht angegeben und es sind keine Achsen-Offsets angegeben.
- *H* ist angegeben doch die angegebene Werkzeugnummer existiert nicht in der Werkzeugtabelle.
- *H* wird angegeben und Achsen werden ebenfalls angegeben.

NOTE G43.2 schreibt nicht in die Werkzeugtabelle.

11.5.33.G43.4 Tool Length Offset on Primary Kinematics

```
G43.4 <H->
```

- *H* - Werkzeugnummer (optional)

G43.4 is *G43* together with a switch to the kinematics the module declares its working transform, so the program runs with tool length compensation in the module's working kinematics. The switch happens first and the offset applies after it, as if the two had been written on consecutive lines. *G49* is the matching cancel: it clears the offset and switches back to identity kinematics, as long as the offset in effect is still *G43.4*'s and no *G12.1* or *G13.1* has selected a kinematics since.

The H word, the offset itself and the parameters it lands in are *G43*'s. Which kinematics is the working one is declared by the module (KINSTYPE_PRIMARY, see the Switchable Kinematics chapter), the number is not the answer; a module that declares none rejects *G43.4* at read time.

The switch is a queue synchronisation point like *G12.1*: when *G43.4* or *G49* changes the kinematics, the interpreter waits for queued motion to finish first, so both stop any blending in progress. Already on the target kinematics they do not switch and blend normally. A plain *G43*, *G43.1* or *G43.2* never switches, whatever kinematics is selected, and the *G49* that cancels one of them does not switch either.

On a machine without switchable kinematics there is nothing to switch: *G43.4* is a plain *G43* and *G49* a plain cancel.

Es ist ein Fehler, wenn:

- the kinematics module is switchable but declares no primary kinematics, or
- any of the *G43* error conditions holds.

11.5.34.G49 Werkzeuglängenkompensation aufheben

- *G49* - hebt die Werkzeuglängenkompensation auf

G49 also switches a switchable kinematics module back to its identity kinematics when it cancels a *G43.4*, undoing the switch that made. It leaves a kinematics selected by *G12.1* or *G13.1* alone, as it does after a plain *G43*, and a module that declares no identity kinematics gets the plain cancel.

Es ist in Ordnung, mit demselben bereits verwendeten Versatz zu programmieren. Es ist auch in Ordnung, ohne Werkzeuglängenkorrektur zu programmieren, wenn gerade keine verwendet wird.

11.5.35.G52 Versatz des lokalen Koordinatensystems

```
G52-Achsen
```

G52 is used in a part program as a temporary "local coordinate system offset" within the workpiece coordinate system. For more information about G92 and G52 and how they interact see Local and Global Offsets. Inside a tilted work plane G52 shifts the plane along its own axes instead.

11.5.36.G53 Bewegen in Maschinenkoordinaten

```
G53 Achsen
```

Um im Maschinenkoordinatensystem zu verfahren, programmieren Sie *G53* auf der gleichen Zeile wie eine lineare Bewegung. *G53* ist nicht modal und muss auf jeder Zeile programmiert werden. *G0* oder *G1* muss nicht auf der gleichen Zeile programmiert werden, wenn eine gerade aktiv ist.

Zum Beispiel *G53 G0 X0 Y0 Z0* bewegt die Achsen zu ihrem Referenzpunkt (die Ausgangsposition), auch wenn das aktuell gewählte Koordinatensystem gültige Offsets hat.

G53-Beispiel

```
G53 G0 X0 Y0 Z0 (Eilgangbewegung zum Maschinenursprung)
G53 X2 (Eilgangbewegung zur absoluten Koordinate X2)
```

Siehe Abschnitt G0 für weitere Informationen.

Es ist ein Fehler, wenn:

- G53 wird verwendet, ohne dass G0 oder G1 aktiv sind,
- oder G53 verwendet wird, während die Fräserkompensation eingeschaltet ist.

11.5.37.G54-G59.3 Koordinatensystem auswählen

- *G54* - Koordinatensystem 1 auswählen
- *G55* - Koordinatensystem 2 auswählen
- *G56* - Koordinatensystem 3 auswählen
- *G57* - Koordinatensystem auswählen 4
- *G58* - Koordinatensystem 5 auswählen
- *G59* - Koordinatensystem 6 auswählen
- *G59.1* - Koordinatensystem 7 auswählen
- *G59.2* - Koordinatensystem 8 auswählen

- *G59.3* - Koordinatensystem 9 auswählen

Die Koordinatensysteme speichern die Achsenwerte und den XY-Drehwinkel um die Z-Achse in den in der folgenden Tabelle aufgeführten Parametern.

Table 90. Koordinatensystem-Parameter

Wählen Sie	CS	X	Y	Z	A	B	C	U	V	W	R
G54	1	5221	5222	5223	5224	5225	5226	5227	5228	5229	5230
G55	2	5241	5242	5243	5244	5245	5246	5247	5248	5249	5250
G56	3	5261	5262	5263	5264	5265	5266	5267	5268	5269	5270
G57	4	5281	5282	5283	5284	5285	5286	5287	5288	5289	5290
G58	5	5301	5302	5303	5304	5305	5306	5307	5308	5309	5310
G59	6	5321	5322	5323	5324	5325	5326	5327	5328	5329	5330
G59.1	7	5341	5342	5343	5344	5345	5346	5347	5348	5349	5350
G59.2	8	5361	5362	5363	5364	5365	5366	5367	5368	5369	5370
G59.3	9	5381	5382	5383	5384	5385	5386	5387	5388	5389	5390

Es ist ein Fehler, wenn:

- Die Auswahl eines Koordinatensystems wird verwendet, wenn die Fräskompensation eingeschaltet ist.

Einen Überblick über Koordinatensysteme finden Sie im Abschnitt Koordinatensystem.

11.5.38.G61 Exakter Pfad Modus

- G61' - Exakter Pfadmodus, Bewegung genau wie programmiert. Die Bewegungen werden nach Bedarf verlangsamt oder gestoppt, um jeden programmierten Punkt zu erreichen. Wenn zwei aufeinanderfolgende Bewegungen exakt kolinear sind, wird die Bewegung nicht angehalten.

11.5.39.G61.1 Exakter Stopp-Modus

- G61.1' - Exakter Stoppmodus, die Bewegung wird am Ende jedes programmierten Segments angehalten.

11.5.40.G64 Pfadübergang

```
G64 <P- <Q-> <R->>
```

- *P* - Toleranz für Bewegungs-Übergänge
- *Q* - naive Nockentoleranz
- *R* - trajectory planner selection and cornering aggressiveness (this fork only)
- *G64* - bestmögliche Geschwindigkeit. Ohne P (oder einen voreingestellten Wert in RS274NGC) bedeutet es, die bestmögliche Geschwindigkeit zu halten, egal wie weit man vom programmierten Punkt entfernt ist.
- G64 P-' - Abwägung zwischen bester Geschwindigkeit und Abweichungstoleranz
- *G64 P- <Q- >* Überlagerung mit Toleranz. Es ist eine Möglichkeit, Ihr System fein abzustimmen, um den besten Kompromiss zwischen Geschwindigkeit und Genauigkeit zu erzielen. Die P-Toleranz bedeutet, dass der tatsächliche Pfad nicht mehr als P- vom programmierten Endpunkt entfernt ist. Die Geschwindigkeit wird bei Bedarf reduziert, um den Pfad beizubehalten. Wenn Sie Q auf einen Wert ungleich Null setzen, wird der *Naive CAM Detector* aktiviert: Wenn es eine Reihe von linearen XYZ-Vorschubbewegungen mit demselben feed rate gibt, die weniger als Q- von der Kollinearität entfernt sind, werden sie zu einer einzigen linearen Bewegung zusammengebrochen. Auf G2/G3 bewegt sich in der G17 (XY) Ebene, wenn die maximale Abweichung eines Bogens von einer geraden Linie kleiner als die G64 P-Toleranz ist, wird der Bogen in zwei Linien aufgeteilt (vom Bogenanfang zum Mittelpunkt und vom Mittelpunkt zum Ende). Diese Zeilen unterliegen dann dem naiven CAM-Algorithmus für Linien. So profitieren Line-Arc-, Arc-Arc- und Arc-Line-Gehäuse sowie Line-Line vom "Naive CAM Detector". Dies verbessert die Konturierungsleistung, indem der Pfad vereinfacht wird. Es ist in Ordnung, für den Modus zu programmieren, der bereits aktiv ist. Weitere Informationen zu diesen Modi finden Sie auch im Abschnitt Trajectory Control. Wenn Q nicht angegeben ist, hat es das gleiche Verhalten wie zuvor und verwendet den Wert von P-. Setzen Sie Q auf Null, um den *Naive CAM Detector* zu deaktivieren.

G64 R: planner selection

The optional R word selects the trajectory planner and its cornering aggressiveness in one modal dial:

- *R omitted* - the planner is left exactly as it was (modal).
- *R0* (or negative) - trapezoidal planner: acceleration is bounded but can step instantly (jerk is not limited).
- *R0.1 .. R1.0* - jerk-limited (S-curve) planner. The value sets the cornering aggressiveness as the fraction of the jerk-feasible corner speed that the look-ahead may use: *R1.0* is full corner speed, lower values corner more gently (*R0.5* matches the initial S-curve release). Values outside 0.1..1.0 are clamped.

The program only requests *trapezoidal* or *smooth*; which smooth planner implements the request is machine configuration ([TRAJ]SMOOTH_PLANNER), so part programs remain valid when the machine's planner implementation changes.

NOTE	R0 to R0.1 is a regime flip, not the gentle end of a gradient: the trapezoidal planner is the

> jerk-unlimited, most aggressive setting, while R0.1 is the gentlest (jerk-limited and slowest through corners). Within R0.1..R1.0 the motion is equally smooth in the jerk sense; only the corner speed changes.

Behavior details:

- A planner switch requested while queued motion remains is latched and applied automatically once no queued motion is left; a message informs the operator and motion is never aborted. The request persists until applied or superseded. Set *G64 ... R* in the program preamble, where the switch is instant.
- If the machine has no smooth planner available (*SMOOTH_PLANNER = 0* or no usable jerk limit configured), *R > 0* is refused with an operator error instead of being silently ignored.
- The R word cannot be used on a line that also contains *M19* (the R meanings would be ambiguous); this is an error.

Es empfiehlt sich, in die Präambel jeder G-Code-Datei eine Pfadsteuerungsangabe aufzunehmen.

Es ist ein Fehler, wenn:

- The path control mode is changed (G61, G61.1 or G64) while cutter radius compensation is active.

G64 P- Q- Beispielzeile

```
G64 P0.015 Q0.015 (stellt die Bahnverfolgung so ein, dass sie innerhalb von 0,015 der
tatsächlichen Bahn liegt)
```

Die P- und Q-Werte sind klein gewählt. Üblicherweise sind diese kleiner als die Genauigkeit der Maschine oder die Genauigkeit der gemeinhin hergestellten Teile. Unten sind Beispiele aufgeführt mit externen P- und Q-Werten für ein Verständnis der Funktion von G64.

G64 Ohne Werte

```
G64 (ohne P- and Q- Werte)
```

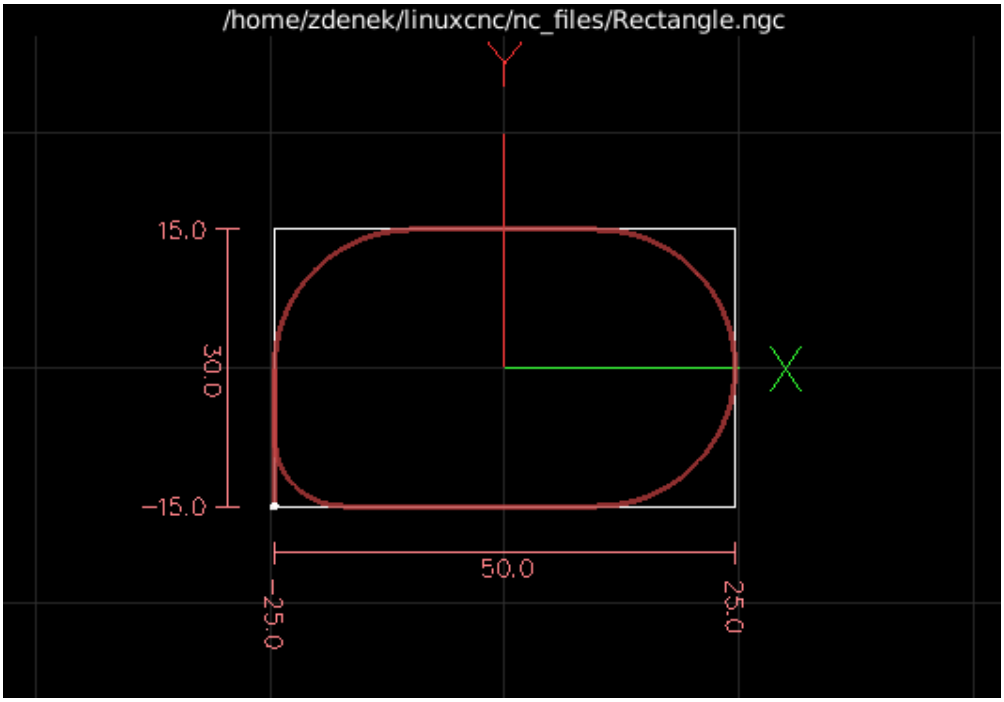


Figure 212. G64 Rechteck

G64 Mit großem Q- Wert

```
G64 P0.015 Q6
```

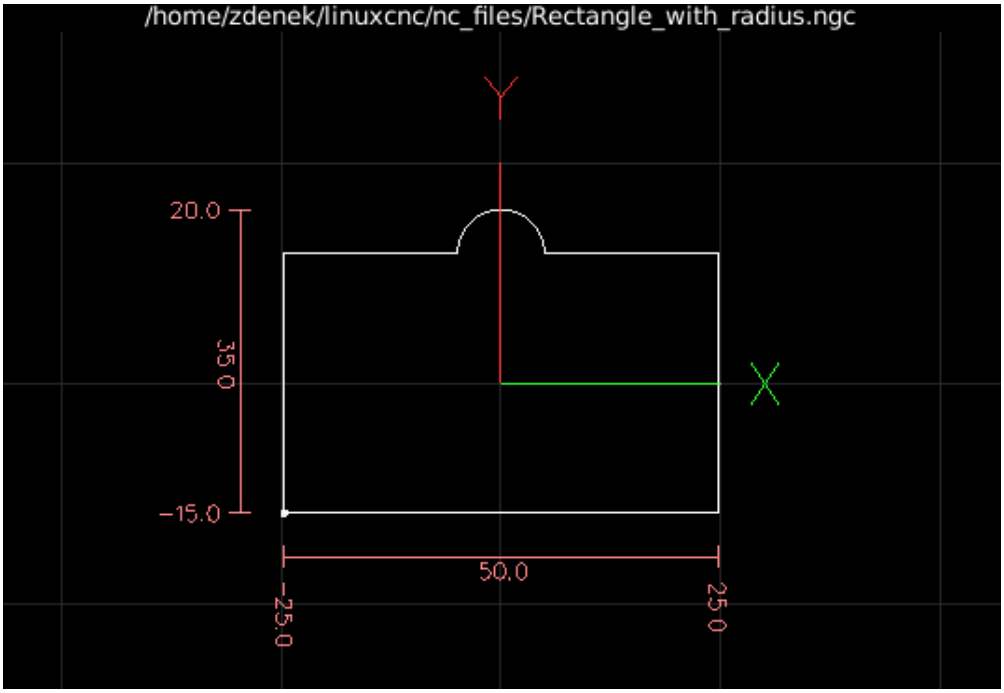


Figure 213. G64 Rechteck mit Radius vor dem Fräsen

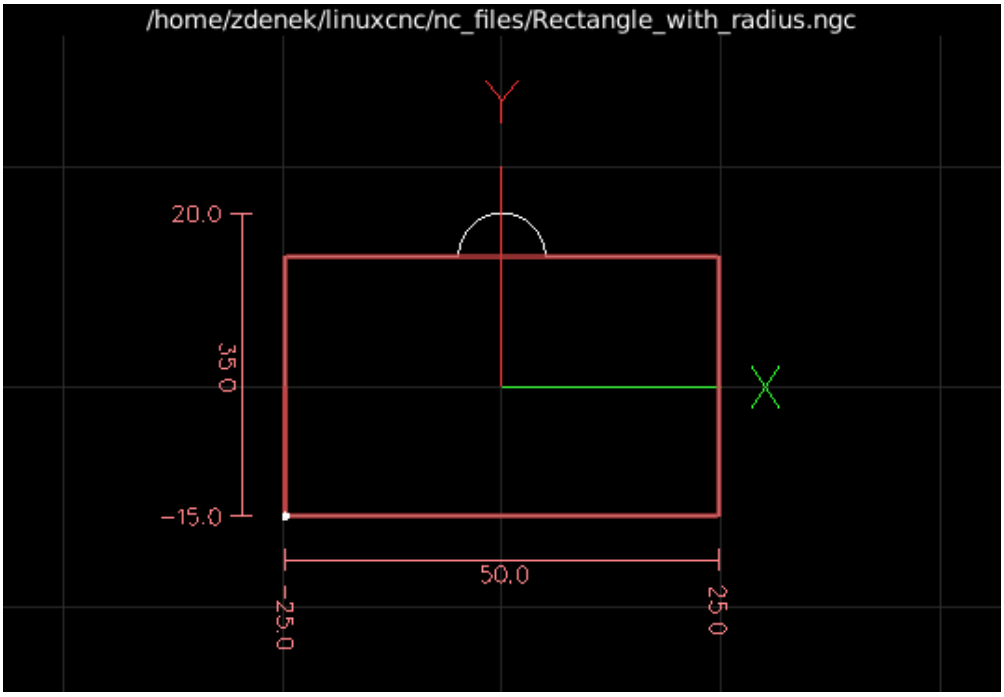


Figure 214. G64 Rechteck mit Radius nach dem Fräsen

```
G64 P0.015 Q2
```

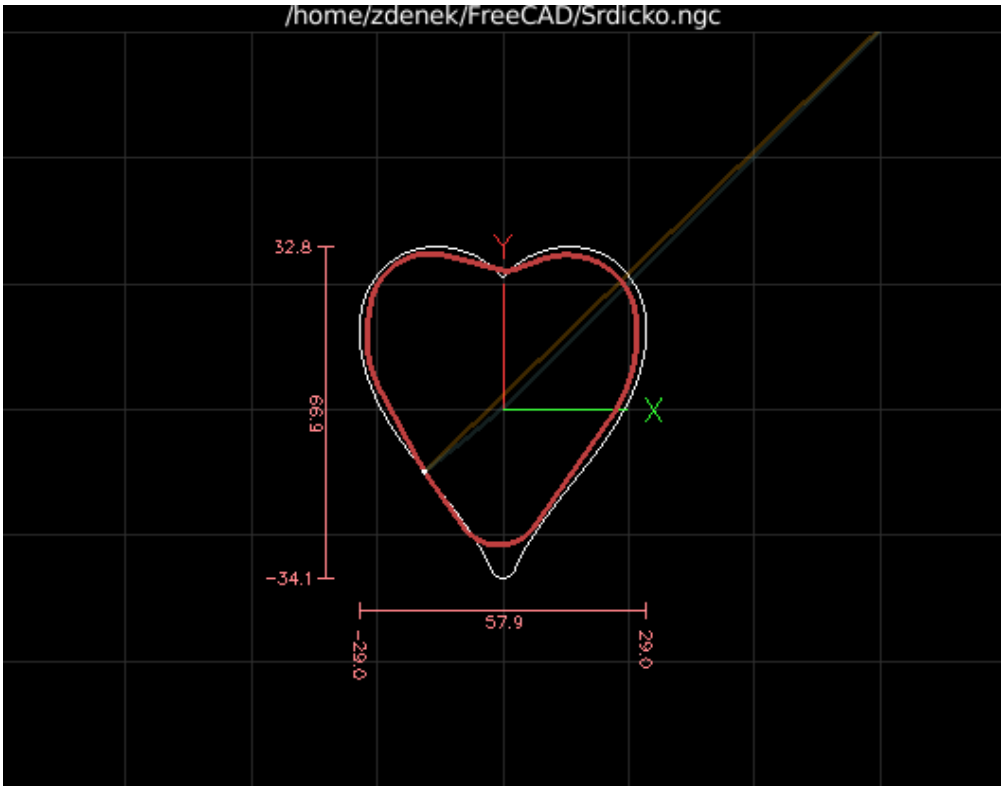


Figure 215. G64 Herz

11.5.41.G68.2, G68.4, G69 Tilted Work Plane

```
G68.2 <P0> <Q-> X- Y- Z- I- J- K- <R->   (three angles, Euler)
G68.2 P1 <Q-> X- Y- Z- I- J- K- <R->     (three angles about fixed axes)
G68.2 P2 Q0 X- Y- Z- <R->                (three points: a shift of the origin, then)
```

```
G68.2 P2 Q1 X- Y- Z- <R->               (a first point,)
G68.2 P2 Q2 X- Y- Z- <R->               (a second point on the plane's +X,)
G68.2 P2 Q3 X- Y- Z- <R->               (a third point on its +Y side; R on one block)
G68.2 P3 Q1 X- Y- Z- I- J- K- <R->      (two vectors: the origin and +X, then)
G68.2 P3 Q2 I- J- K-                    (+Z, the normal)
G68.4 ...                               (any G68.2 form, on the active plane)
G69                                     (cancel)
```

A tilted work plane is a coordinate system composed on top of the active one: the blocks between the definition and *G69* are programmed in the plane, with X and Y in it and Z along its normal, while the work offset underneath, *G54* say, is untouched. Positions on the display, probe results and G53 all take the plane into account. It is the same offset chain as always with one more stage, applied first:

```
absolute = tool offset + G5x + XY rotation applied to (G92 + origin + rotation applied to
(G52 in the plane + program))
```

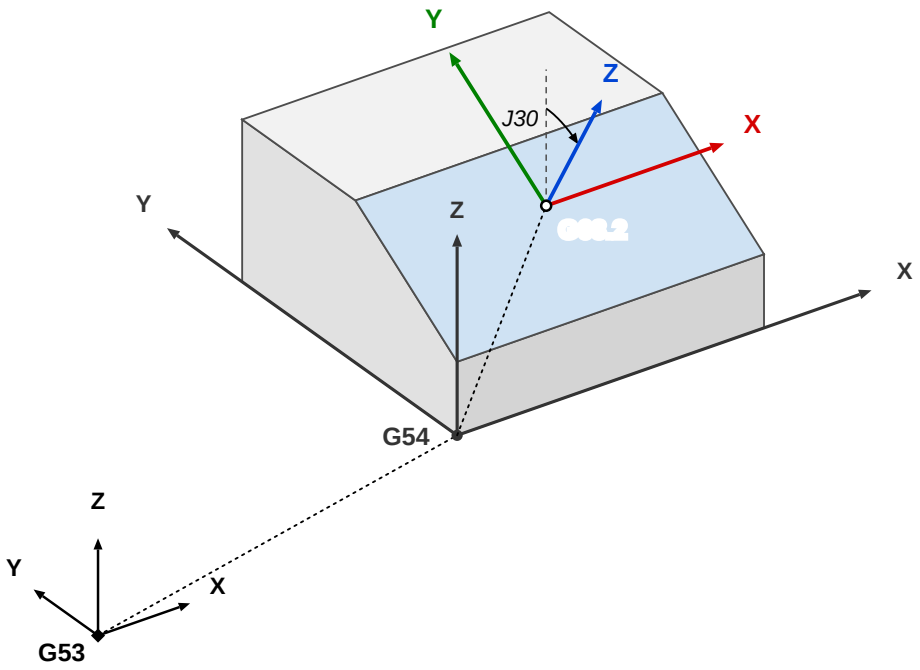


Figure 216. G68.2 X50 Y30 Z40 J30: a plane tilted 30 degrees about X, on G54, on the machine coordinates

X, *Y* and *Z* are the plane's origin (*P2* gives it by points, below) and the plane's rotation is built from the rest of the words, both in the coordinate system that is active when the plane is defined: the work offset with *G92* and the XY rotation in place, which is what the position display shows at that moment. A rotary word does not pass through the plane, since on a TCP kinematics the rotary coordinates are the rotary joints and a plane does not change what a joint is. Words left out are zero.

P selects how the rotation is given:

- *P0*, or no *P*: three angles *I*, *J*, *K* applied one after another, each about an axis of the plane as rotated so far (Euler angles). *Q* names the axes with three digits, 1 for X, 2 for Y and 3 for Z, no two adjacent alike; the default is *Q313*, Z then X then Z.
- *P1*: three angles *I*, *J*, *K*, each about an axis of the coordinate system the plane is defined in, in the order *Q* gives; the default is *Q123*, X then Y then Z.

- *P2*: three points, over up to four blocks with *Q0* to *Q3*. The direction from the first point to the second is the plane's +X, the third point lies on the +Y side. The origin is the first point. The *Q0* block, which may be left out, moves the origin by its *X*, *Y*, *Z* along the plane's own axes, after *R*: *Q0 X10* puts the origin 10 along the plane's X from the first point, the way *X10* in a program under the plane would. *R* may be given on any one of the blocks.
- *P3*: two vectors, over two blocks. *Q1* gives the origin and the +X direction in *I*, *J*, *K*; *Q2* gives +Z, the normal, in *I*, *J*, *K*. The X direction is kept as given. The normal need not be exactly at right angles to it, within 5 degrees; the part of the normal along X is dropped.

R turns the plane about its own Z after everything else, in degrees.

The forms follow the Fanuc tilted working plane commands, with a few extensions: *R* on *P0*, *P1* and *P3* (Fanuc gives it with *P2* only), *Q* on *P0* (Fanuc's Euler angles are always Z, X, Z), *P1* orders that repeat an axis, such as *Q121*, besides the six that name each axis once, and all three angles zero, which is the plane of the coordinate system underneath rather than an error, as on a Fanuc with parameter ATW set.

The blocks of a *P2* or *P3* definition have to follow one another; any other block in between is an error. A definition left unfinished in MDI is dropped when a program is opened. A definition with a plane already active replaces it, with the words in the coordinate system underneath, not in the old plane.

G68.4 takes any *G68.2* form and composes it onto the active plane: the words are in the plane, and the result is a new plane relative to the old one. It needs a plane to build on.

The G-code preview draws every plane a program defines, so a program can be checked for where its planes sit before it runs: a rectangle lying in the plane, over the moves made under it with a margin around them, the plane's own X, Y and Z at the centre of the rectangle in the colours of the machine axes, and a cross at the plane's origin, which need not lie in the rectangle: a program drilling a sphere puts every plane's origin at the centre and works out on the surface. The rectangle is drawn at the plane's Z where the program comes down to it, and otherwise at the end of the Z range the program worked in that is nearest to it, so a drilling cycle that stays above the origin shows its holes where they are cut; a plane nothing moved under is drawn as a square sized from the program. A plane restated with the same words is one plane, not one per restatement. The plane in effect on the machine, the one the last executed *G68.2* set, is drawn over the others in its own colour, so as the program runs the plane it is in stands out from the ones it has been in and will be in; a plane set from MDI shows the same way, as a square with nothing under it. The plane in effect is drawn no smaller than the coordinate system axes, with its own axes as long as those, so its orientation can be checked against them before a program runs, however small the program.

G69 cancels the plane. So do the end of the program, *M2* or *M30*, and an abort, unless the INI setting `[RS274NGC]RETAIN_WORK_PLANE = 1` keeps it through both. A plane set up in MDI then serves every run of a program that does not set one itself. With the setting on:

- After an abort the plane is the one the machine was in when it stopped, not one the interpreter had read further on. If the work offset under it is not the one the interpreter returns to, the plane is cancelled.
- At the end of a program the plane stays where it is in the coordinate system, and the coordinate system changes under it as the end of a program changes it: to *G54* unless `DISABLE_AUTO_G54 = 1`, and without *G92* if `DISABLE_G92_PERSISTENCE = 1`.

- An `ON_ABORT_COMMAND` runs inside the plane; a change of coordinate system there takes the plane along, as below.

Nothing about the plane is written to the parameter file, so a restart always cancels it.

Defining the plane does not move anything.

While a plane is active the codes that change the coordinate system the plane sits on are refused: *G92*, *G92.1*, *G92.2*, *G92.3*, *G10 L2*, *G10 L20*, *G10 L10* and *G10 L11*. Cancel the plane first.

A change of coordinate system (*G54* to *G59.3*) in a plane takes the plane along: its origin and rotation stay the same relative to the coordinate system, so the plane moves with the work offset and turns with its XY rotation, as on Fanuc with parameter 3TW set. The same tilted feature on several fixtures is one *G68.2* and a *G55*, *G56*, ... before each.

G52 in a plane takes *X*, *Y* and *Z* only, and shifts the plane along its own axes, as Fanuc's local coordinate system does in a tilted working plane: *G52 X10* moves the program's zero 10 along the plane's X. It is not written to the *G92* parameters; *G52 X0 Y0 Z0* takes it out, and *G69* drops it. A *G52* given before the plane stays in *G92*, under the plane. A new plane, by *G68.2* or *G68.4*, is refused while a *G52* in the plane is in effect, as on Fanuc.

The active plane is reported in the modal G-code display as the code that defined it. Status carries it as `g68_offset`, `g68_rotation` and `g68_active`, next to the other offsets, for displays that want to show plane coordinates or draw the plane.

G68.2 Example

```
G54
G68.2 X50 Y50 Z0 I30 J20 K0     (Euler: 30 about Z, 20 about the new X)
G0 X0 Y0 Z5                      (5 above the plane's origin, along its normal)
G1 Z-3 F150                      (a hole 3 deep, straight into the plane)
G0 Z5
G68.4 X20 P1 I0 J0 K90           (a plane 20 along X in the old one, turned 90 about Z)
G0 X0 Y0 Z5
G69                              (back to G54 as it was)
```

Es ist ein Fehler, wenn:

- *P* is not 0, 1, 2 or 3, or *Q* with *P0* or *P1* is not three axis digits with no two adjacent alike.
- A *P2* or *P3* definition is interrupted, or its *Q* words come out of order.
- The points of a *P2* definition coincide or lie on one line, or a vector of a *P3* definition is zero or the two are 5 degrees or more off square.
- A word the form does not read is given: *I*, *J* or *K* with *P2*, *X*, *Y*, *Z* or *R* on the *P3 Q2* block, or *A*, *B*, *C*, *U*, *V* or *W* with any form.
- *R* is given on more than one block of a *P2* definition.
- *G68.4* is used with no plane active.
- *G68.2* or *G68.4* is used while a *G52* given in the plane is in effect.
- Cutter compensation is on.

- Polar coordinates or a motion code are used on the same line.

11.5.42.G70 Endbearbeitungszyklus der Drehmaschine

```
G70 Q- <X-> <Z-> <D-> <E-> <P->
```

- *Q* - Die Unterroutinenummer.
- *X* - Die Anfangsposition X, standardmäßig die Ausgangsposition.
- *Z* - Die Startposition Z ist standardmäßig auf die Ausgangsposition eingestellt.
- *D* - Der Startabstand des Profils ist standardmäßig auf 0 eingestellt.
- *E* - Der Endabstand des Profils ist standardmäßig auf 0 eingestellt.
- *P* - Die Anzahl der zu verwendenden Durchläufe ist standardmäßig 1.

Der Zyklus *G70* soll verwendet werden, nachdem die im Unterprogramm mit der Nummer Q angegebene Form des Profils mit G71 oder G72 geschnitten wurde.

- Vorbereitende Bewegungen (engl. preliminary motion).
  - Wenn Z oder X verwendet werden, wird ein Eilgang zu dieser Position ausgeführt. Diese Position wird auch zwischen den einzelnen Finishing-Durchgängen verwendet.
  - Dann wird ein Eilgang an den Anfang des Profils ausgeführt.
  - Der in Q- angegebene Pfad wird mit den Befehlen G1 und G2, G3 Bogenbewegung verfolgt.
  - Wenn ein weiterer Durchgang erforderlich ist, erfolgt ein weiterer Eilgang zur Zwischenposition, bevor ein Eilgang zum Anfang des Profils durchgeführt wird.
  - Nach dem letzten Durchgang bleibt das Werkzeug am Ende des Profils einschließlich E- stehen.
- Mehrere Durchgänge. Der Abstand zwischen dem Durchgang und dem endgültigen Profil ist (Durchgang-1)*(D-E)/P+E. Dabei ist pass die Nummer des Durchgangs und D, E und P sind die Nummern D/E/P.
- Der Abstand wird anhand der Startposition des Zyklus berechnet, wobei der Abstand zu diesem Punkt positiv ist.
- Verrundungen und Fasen im Profil. Es ist möglich, Verrundungen oder Fasen in das Profil einzufügen, siehe G71 G72 Schruppzyklus der Drehmaschine für weitere Details.

Es ist ein Fehler, wenn:

- Es ist kein Unterprogramm mit der in Q angegebenen Nummer definiert.
- Der im Profil angegebene Weg ist nicht monoton in Z oder X.
- G17 - G19.1 Ebene auswählen wurde nicht zur Auswahl der ZX-Ebene verwendet.

11.5.43.G71 G72 Schruppzyklus der Drehmaschine

NOTE	Die G71 und G72 Zyklen sind derzeit etwas fragil. Siehe zum Beispiel Problem-Meldung

(engl. *issue*) #2939.

```
G71   Q- <X-> <Z-> <D-> <I-> <R->
G71.1 Q- <X-> <Z-> <D-> <I-> <R->
G71.2 Q- <X-> <Z-> <D-> <I-> <R->
G72   Q- <X-> <Z-> <D-> <I-> <R->
G72.1 Q- <X-> <Z-> <D-> <I-> <R->
G72.2 Q- <X-> <Z-> <D-> <I-> <R->
```

- *Q* - Die Unterroutinenummer.
- *X* - Die Anfangsposition X, standardmäßig die Ausgangsposition.
- *Z* - Die Startposition Z ist standardmäßig auf die Ausgangsposition eingestellt.
- *D* - Der verbleibende Abstand zum Profil ist standardmäßig auf 0 eingestellt.
- *I* - Das Schnittinkrement, standardmäßig 1.
- *R* - Der Rückzugsabstand, standardmäßig 0,5.

Der Zyklus G71/G72 ist für das Schruppen eines Profils auf einer Drehmaschine vorgesehen. Die G71-Zyklen entfernen Schichten des Materials, während sie in Z-Richtung verfahren. Die G72-Zyklen tragen Material ab, während sie in der X-Achse verfahren, der so genannte Plandrehzyklus. Die Verfahrrichtung ist die gleiche wie bei dem im Unterprogramm angegebenen Weg. Für den Zyklus G71 muss sich die Z-Koordinate monoton ändern, für den Zyklus G72 ist dies für die X-Achse erforderlich.

Das Profil wird in einer Unterroutine mit der Nummer Q- angegeben. Dieses Unterprogramm kann die Bewegungsbefehle G0, G1, G2 und G3 enthalten. Alle anderen Befehle werden ignoriert, einschließlich Vorschub- und Geschwindigkeitseinstellungen. Die G0 Eilgang Bewegung Befehle werden als G1 Befehle interpretiert. Jeder Bewegungsbefehl kann auch eine optionale A- oder C- Nummer enthalten. Wird die Zahl A- hinzugefügt, so wird am Endpunkt der Bewegung eine Verrundung mit dem durch A angegebenen Radius eingefügt; wenn dieser Radius zu groß ist, schlägt der Algorithmus mit einem nicht monotonen Pfadfehler fehl. Es ist auch möglich, die C-Nummer zu verwenden, wodurch eine Fase eingefügt werden kann. Diese Fase hat die gleichen Endpunkte wie eine Verrundung mit den gleichen Abmessungen, aber es wird eine gerade Linie anstelle eines Bogens eingefügt.

Im absoluten Modus können U (für X) und W (für Z) als inkrementelle Verschiebungen verwendet werden.

Die G7x.1-Zyklen schneiden keine Taschen. Die G7x.2-Zyklen schneiden nur nach der ersten Tasche und machen dort weiter, wo G7x.1 aufgehört hat. Es ist ratsam, vor dem G7x.2-Zyklus etwas zusätzliches Material zum Schneiden übrig zu lassen. Wenn also G7x.1 einen D1.0-Zyklus verwendet hat, kann G7x.2 einen D0.5-Zyklus verwenden und 0,5 mm werden beim Übergang von einer Tasche zur nächsten entfernt.

Die normalen G7x-Zyklen schneiden das gesamte Profil in einem Zyklus.

1. Vorbereitende Bewegungen (engl. preliminary motion).
   - Wenn Z oder X verwendet werden, wird ein rapid move zu dieser Position ausgeführt.
   - Nach dem Schneiden des Profils hält das Werkzeug am Ende des Profils an, einschließlich des in

D angegebenen Abstands.

2. Die D-Nummer wird verwendet, um einen Abstand zum endgültigen Profil einzuhalten, damit Material für die Nachbearbeitung übrig bleibt.

Es ist ein Fehler, wenn:

- Es ist kein Unterprogramm mit der in Q angegebenen Nummer definiert.
- Der im Profil angegebene Weg ist nicht monoton in Z oder X.
- G17 - G19.1 Ebene auswählen wurde nicht zur Auswahl der ZX-Ebene verwendet.
- G41 G42 Fräserkompensation ist aktiv.

11.5.44.G73 Bohrzyklus mit Spanbruch

```
G73 X- Y- Z- R- Q- D- <L->
```

- *R* - Rückzugsposition entlang der Z-Achse.
- *Q* - Delta-Inkrement entlang der Z-Achse.
- *L* - wiederholen

The *G73* cycle is drilling or milling with chip breaking. This cycle takes a Q number which represents a *delta* increment along the Z axis. Peck clearance can be specified by optional D number.

- Vorbereitende Bewegungen (engl. preliminary motion).
  - Wenn die aktuelle Z-Position unter der R-Position liegt, führt die Z-Achse eine schnelle Bewegung in die R-Position aus.
  - Bewegen zu den X-Y-Koordinaten
- Bewege die Z-Achse nur mit dem aktuellen Vorschub nach unten um Delta oder auf die Z-Position, je nachdem, was weniger tief ist.
- Rapid up either the D value, the G73_PECK_CLEARANCE specified in the INI file or the default of .010" / 0.254 mm.
- Wiederholen der Schritte 2 und 3, bis die Z-Position bei Schritt 2 erreicht ist.
- Die Z-Achse fährt im Eilgang in die R-Position.

Es ist ein Fehler, wenn:

- Die Q-Zahl ist negativ oder null.
- die R-Nummer ist nicht angegeben

11.5.45.G74 Linkshändiger Gewindeschneidzyklus mit Verweilzeit

```
G74 (X- Y- Z-) oder (U- V- W-) R- L- P- $- F-
```

- *R-* - Zurückziehen der Position entlang der Z-Achse.
- L-' - Wird im inkrementellen Modus verwendet; Anzahl der Wiederholungen des Zyklus. Siehe G81 für Beispiele.
- P-' - Verweilzeit (Sekunden).
- *$-* - Ausgewählte Spindel.
- *F-* - Vorschubgeschwindigkeit (Spindeldrehzahl multipliziert mit der pro Umdrehung zurückgelegten Strecke (Gewindesteigung)).

WARNING	G74 verwendet keine synchronisierte Bewegung.

Der *G74* Zyklus ist für das Gewindeschneiden mit schwimmendem Spannfutter und Verweilzeit am Bohrungsgrund vorgesehen.

1. Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
2. Deaktivieren von Vorschub- und Geschwindigkeits-Neufestsetzungen (engl. overrides).
3. Fahren Sie die Z-Achse mit der aktuellen Vorschubgeschwindigkeit in die Z-Position.
4. Anhalten der ausgewählten Spindel (ausgewählt durch den Parameter $)
5. Drehen der Spindel im Uhrzeigersinn.
6. Verweilen für die Anzahl von P Sekunden.
7. Bewegen Sie der Z-Achse mit der aktuellen Vorschubgeschwindigkeit, um Z zu löschen
8. Wiederherstellung der Vorschub- und Geschwindigkeitsneufestsetzung-Aktivierung in den vorherigen Zustand

Die Länge der Verweilzeit wird durch ein *P*--Wort im G74-Satz angegeben. Die Vorschubgeschwindigkeit *F-* ist die Spindeldrehzahl multipliziert mit dem Abstand pro Umdrehung (Gewindesteigung). Im Beispiel S100 mit 1,25mm pro Umdrehung Gewindesteigung ergibt einen Vorschub von F125.

11.5.46.G76 Gewindedrehen-Zyklus (engl. threading)

```
G76 P- Z- I- J- R- K- Q- H- E- L- $- D-
```

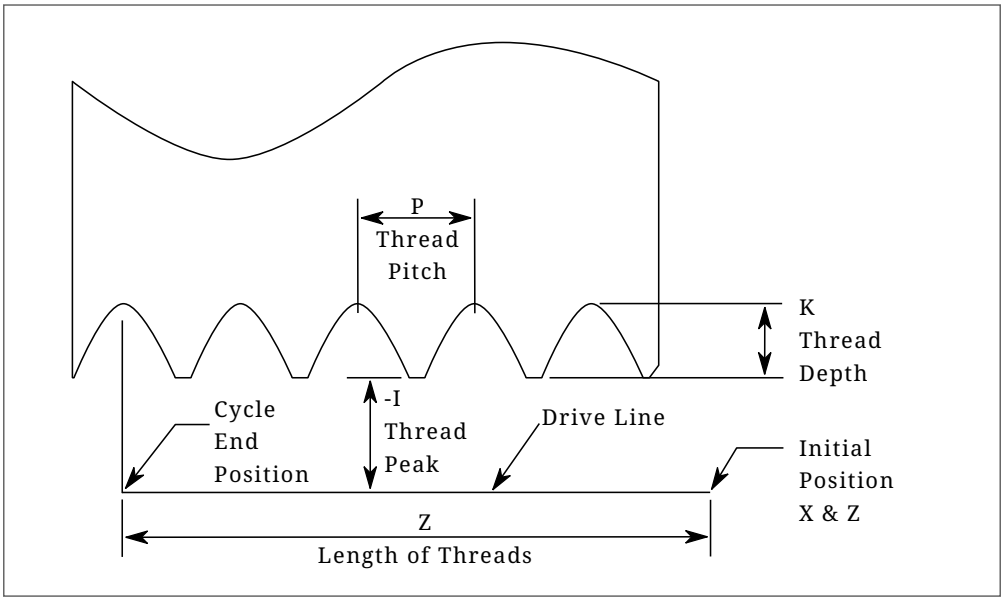


Figure 217. G76 Gewindeschneiden

- *Drive Line* - Eine Linie durch die anfängliche X-Position parallel zum Z.
- *P-* - Die *Gewindesteigung* in Abstand pro Umdrehung.
- *Z-* - Die endgültige Position von Windungen. Am Ende des Zyklus befindet sich das Werkzeug an dieser Z-Position.

NOTE Wenn G7 *Drehmaschinen Durchmesser Modus* (engl. Lathe Diameter Mode') aktiv ist, sind die Werte für *I, J* und *K* Durchmessermessungen. Wenn G8 *Drehradiusmodus* in Kraft ist, sind die Werte für *I, J* und *K* Radiusmessungen.

- *I-* - Die *Gewindespitze* (engl. thread peak), die von der *Antriebslinie* (engl. drive line) versetzt ist. Negative *I*-Werte sind Außengewinde und positive *I*-Werte sind Innengewinde. Im Allgemeinen wurde das Material vor dem *G76*-Zyklus auf diese Größe gedreht.
- *J-* - Ein positiver Wert, der die "anfängliche Schnitttiefe" angibt. Der erste Gewindeschnitt liegt *J* hinter der *Gewindespitzen*-Position.
- ' K-' - Ein positiver Wert, der die *volle Gewindetiefe* angibt. Der endgültige Gewindeschnitt liegt *K* über der *Gewindespitzenposition*.

Optionale Einstellungen

- *$-* - Die Spindelnummer, mit der die Bewegung synchronisiert werden soll (Standardwert 0). Wird z.B. $1 programmiert, beginnt die Bewegung mit dem Reset von `spindle.1.index-enable` und verläuft synchron mit dem Wert von `spindle.1.revs`.
- *R-* - Die *Tiefendegression. R1.0* wählt eine konstante Tiefe bei aufeinanderfolgenden Einfädelgängen. *R2.0* wählt eine konstante Fläche. Werte zwischen 1,0 und 2,0 wählen eine abnehmende Tiefe bei zunehmender Fläche. Werte über 2,0 wählen eine abnehmende Fläche. Beachten Sie, dass unnötig hohe Degressionswerte dazu führen, dass eine große Anzahl von Durchgängen verwendet wird. (Degression = ein stufenweiser Abstieg)

WARNING Unnötig hohe Degressionswerte führen zu einer unnötig hohen Anzahl von

Durchgängen. (Degression = Tauchen in Stufen)

- *Q-* - Der *zusammengesetzte Gleitwinkel* ist der Winkel (in Grad), der beschreibt, inwieweit aufeinanderfolgende Durchgänge entlang der Antriebslinie versetzt sein sollten. Dies wird verwendet, um eine Seite des Werkzeugs zu veranlassen, mehr Material als die andere zu entfernen. Ein positiver *Q*-Wert bewirkt, dass die Vorderkante des Werkzeugs stärker schneidet. Typische Werte sind 29, 29,5 oder 30.
- ' H-' - Die Anzahl der "Frühjahrsdurchgänge" (engl. spring/finishing passes). Solche abschließenden Durchgänge sind zusätzliche Durchgänge bei voller Gewindetiefe. Wenn keine zusätzlichen Durchgänge gewünscht sind, programmieren Sie *H0*.
- *D-* - The *start angle offset* in degrees from the spindle index pulse. Every pass of the cycle begins once the spindle has turned *D* degrees past the index pulse, instead of at the index pulse itself. If *D* is omitted the default is zero, which starts every pass at the index pulse as before. This cuts one start of a multi-start thread: run the same *G76* cycle once per start, each with a different *D*. A two-start thread is cut with *D0* and then *D180*, a three-start thread with *D0*, *D120* and *D240*.

NOTE *D* is an angle past the index pulse and so is only meaningful as a positive number. A negative value is not an error: its magnitude is used, so *D-90* behaves exactly as *D90*. Values of 360 or more are not reduced, so *D450* waits a full turn longer than *D90*.

Die Gewindeein- und -ausgänge können mit den Werten "E" und "L" konisch programmiert werden.

- *E-* - Gibt den Abstand entlang der Antriebslinie an, der für die Verjüngung verwendet wird. Der Winkel der Verjüngung ist so, dass sich der letzte Durchgang über die mit E angegebene Strecke zum Gewindescheitel verjüngt. *E0.2* ergibt eine Verjüngung für die ersten/letzten 0,2 Längeneinheiten entlang des Gewindes. Für eine 45 Grad Verjüngung programmieren Sie E wie K.
- L-' - Gibt an, welche Enden des Gewindes die Verjüngung erhalten. Programmieren Sie *L0* für keine Verjüngung (der Ausgangswert), *L1* für Eingangsverjüngung, *L2* für Ausgangsverjüngung oder *L3* für Eingangs- und Ausgangsverjüngung. Einlaufkegel halten an der Antriebslinie an, um sich mit dem Indeximpuls zu synchronisieren, und bewegen sich dann mit feed rate zum Anfang des Kegels. Ohne Einfahrkegel fährt das Werkzeug im Eilgang auf die Schnitttiefe, synchronisiert sich und beginnt den Schnitt.

```
Das Werkzeug wird vor der Ausgabe des G76 in die X- und Z-Ausgangsposition gefahren.
Die X-Position ist die "Antriebslinie" und die Z-Position ist der Beginn des Gewindes.
```

Das Werkzeug macht vor jedem Gewindedurchgang eine kurze Synchronisationspause, so dass eine Entlastungsnut am Einlauf erforderlich ist, es sei denn, der Gewindeanfang liegt hinter dem Ende des Materials oder es wird ein Einlaufkegel verwendet.

Wird kein Ausgangskegel verwendet, ist die Ausgangsbewegung nicht mit der Spindeldrehzahl synchronisiert und wird ein Eilgang sein. Bei einer langsamen Spindel kann die Ausfahrbewegung nur einen kleinen Bruchteil einer Umdrehung dauern. Wenn die Spindeldrehzahl nach mehreren Durchgängen erhöht wird, benötigen die nachfolgenden Ausfahrbewegungen einen größeren Teil einer Umdrehung, was zu einem sehr starken Schnitt während der Ausfahrbewegung führt. Dies kann vermieden werden, indem eine Entlastungsnut am Ausgang vorgesehen wird oder indem die

Spindeldrehzahl während des Gewindeschneidens nicht verändert wird.

Die endgültige Position des Werkzeugs befindet sich am Ende der "Antriebslinie". Um das Werkzeug aus der Bohrung zu entfernen, ist eine sichere Z-Bewegung mit einem Innengewinde erforderlich.

Es ist ein Fehler, wenn:

- Die aktive Ebene ist nicht die ZX-Ebene.
- Andere Achsenbezeichnungen wie X- oder Y- werden angegeben.
- Der *R*- Degressionswert ist kleiner als 1,0.
- Es sind nicht alle erforderlichen Angaben enthalten.
- "P-", "J-", "K-" oder "H-" ist negativ.
- "E-" ist größer als die halbe Länge der Antriebslinie.
- The pitch and the spindle speed together ask an axis to move faster than its maximum velocity. See G33 for how the speed is determined in constant surface speed mode.

NOTE	The spindle speed override is held for the whole G76 cycle at the value in effect when the cycle starts, capped at 100% because the pitch is checked against the programmed speed. Moving it while the cycle runs has no effect, so every pass cuts at one speed and they stay on the same helix. The previous setting is restored when the cycle ends. To thread at 80% of the programmed speed, set the override before the cycle starts.

HAL-Verbindungen

Die Pins *spindle.N.at-speed* und *encoder.n.phase-Z* für die Spindel müssen in Ihrer HAL-Datei angeschlossen sein, damit G76 funktioniert. Siehe die Spindel-Pins im Abschnitt Bewegung für weitere Informationen.

Technische Informationen

Der G76 Festzyklus basiert auf der G33 Spindel-Synchronbewegung. Weitere Informationen finden Sie in der G33 Technical Info.

Das Beispielprogramm *g76.ngc* zeigt die Verwendung des G76-Festzyklus und kann auf jeder Maschine mit der Konfiguration *sim/lathe.ini* angezeigt und ausgeführt werden.

G76 Beispielcode

```
G0 Z-0.5 X0.2
G76 P0.05 Z-1 I-.075 J0.008 K0.045 Q29.5 L2 E0.045
```

In der Abbildung befindet sich das Werkzeug in der Endposition, nachdem der G76-Zyklus abgeschlossen ist. Sie sehen rechts den Einfahrweg vom Q29.5 und links den Ausfahrweg vom L2 E0.045. Die weißen Linien sind die Schnittbewegungen.

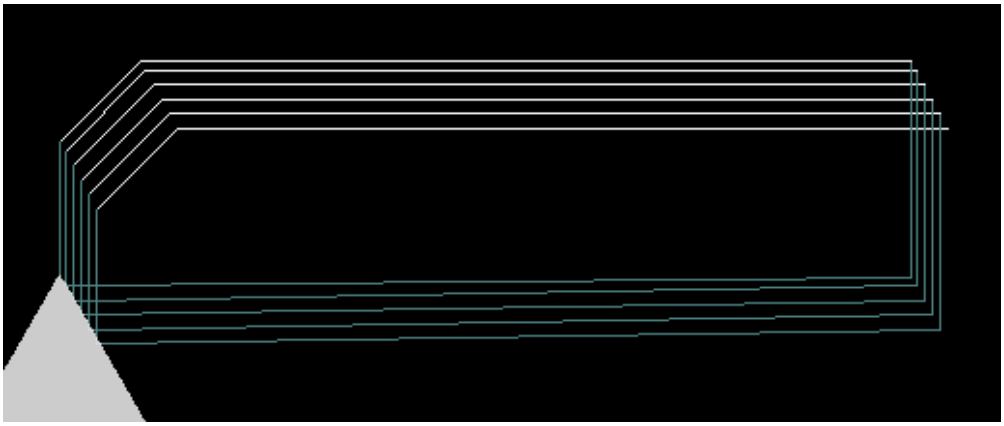


Figure 218. G76 Beispiel

11.5.47.G80-G89 Festzyklen (engl. canned cycles)

In diesem Abschnitt werden die Festzyklen *G81* bis *G89* und der Festzyklusstopp *G80* beschrieben.

Alle Festzyklen werden in Bezug auf die aktuell gewählte Ebene ausgeführt. Jede der neun Ebenen kann ausgewählt werden. In diesem Abschnitt wird bei den meisten Beschreibungen davon ausgegangen, dass die XY-Ebene ausgewählt wurde. Das Verhalten ist analog, wenn eine andere Ebene gewählt wird, und es müssen die richtigen Worte verwendet werden. In der Ebene "G17.1" zum Beispiel verläuft die Wirkung des Festzyklus entlang W, und die Positionen oder Inkremente werden mit U und V angegeben.

Drehachsenbegriffe sind in Festzyklen nicht erlaubt. Wenn die aktive Ebene zur XYZ-Familie gehört, sind die UVW-Achsenwörter nicht erlaubt. Wenn die aktive Ebene zur UVW-Familie gehört, sind die XYZ-Achsenwörter ebenfalls nicht erlaubt.

Geläufige Begriffe

Alle Festzyklen verwenden X-, Y-, Z- oder U-, V-, W-Gruppen, je nach gewählter Ebene und R-Wort. Die R-Position (in der Regel bedeutet sie Rückzug) befindet sich entlang der Achse, die senkrecht zur aktuell gewählten Ebene steht (Z-Achse für XY-Ebene usw.) Einige Festzyklen verwenden zusätzliche Argumente.

Anhaftende Begriffe

Bei Festzyklen bezeichnen wir eine Zahl als "haftend" (engl. sticky), wenn derselbe Zyklus in mehreren aufeinanderfolgenden Codezeilen verwendet wird und die Zahl beim ersten Mal verwendet werden muss, aber in den übrigen Zeilen optional ist. Sticky-Zahlen behalten ihren Wert in den übrigen Zeilen bei, wenn sie nicht ausdrücklich anders programmiert sind. Die R-Nummer ist immer "sticky".

Im inkrementellen Abstandsmodus werden die X-, Y- und R-Zahlen als Inkremente von der aktuellen Position und Z als Inkrement von der Position der Z-Achse behandelt, bevor die Bewegung mit Z stattfindet. Im absoluten Abstandsmodus sind die X-, Y-, R- und Z-Zahlen absolute Positionen im aktuellen Koordinatensystem.

Zyklus wiederholen

Die Angabe L ist optional und gibt die Anzahl der Wiederholungen an. L=0 ist nicht erlaubt. Wird die Wiederholungsfunktion verwendet, dann wird sie normalerweise im inkrementellen Abstandsmodus eingesetzt, so dass dieselbe Bewegungssequenz an mehreren gleichmäßig verteilten Stellen entlang einer geraden Linie wiederholt wird. Wenn L- im inkrementellen Modus bei ausgewählter XY-Ebene größer als 1 ist, werden die X- und Y-Positionen durch Addition der angegebenen X- und Y-Zahlen entweder zu den aktuellen X- und Y-Positionen (beim ersten Durchgang) oder zu den X- und Y-Positionen am Ende des vorherigen Durchgangs (bei den Wiederholungen) bestimmt. Wenn Sie also *L10* programmieren, erhalten Sie 10 Zyklen. Der erste Zyklus ist die Entfernung X,Y von der ursprünglichen Position. Die Positionen R und Z ändern sich während der Wiederholungen nicht. Die L-Nummer ist nicht unveränderlich. Im absoluten Abstandsmodus bedeutet L>1, dass der gleiche Zyklus mehrmals an der gleichen Stelle durchgeführt wird. Das Weglassen des L-Werts ist gleichbedeutend mit der Angabe von L=1.

Rückzugsmodus

Die Höhe der Rückzugsbewegung am Ende jeder Wiederholung (in den folgenden Beschreibungen als *clear Z* bezeichnet) wird durch die Einstellung des Rückzugsmodus bestimmt, entweder auf die ursprüngliche Z-Position (wenn diese über der R-Position liegt und der Rückzugsmodus *G98*, OLD_Z, ist), oder andernfalls auf die R-Position. Siehe den Abschnitt G98 G99.

Festzyklus Fehler

Es ist ein Fehler, wenn:

- alle Achsenwörter während eines Festzyklus fehlen,
- Achsenwörter aus verschiedenen Gruppen (XYZ) (UVW) zusammen verwendet werden,
- eine P-Nummer erforderlich ist und eine negative P-Nummer verwendet wird,
- eine L-Zahl verwendet wird, die nicht als positive ganze Zahl ausgewertet werden kann,
- die Bewegung der Drehachse während eines Festzyklus verwendet wird,
- während eines Festzyklus ist die inverse Zeitvorschubgeschwindigkeit aktiv ist,
- oder Fräserkompensation während eines Festzyklus aktiv ist.

Bei aktiver XY-Ebene aktiv ist die Z-Nummer nicht veränderbar, und es ist ein Fehler, wenn:

- die Z-Nummer fehlt und derselbe Festzyklus nicht bereits aktiv war,
- oder die R-Zahl kleiner ist als die Z-Zahl.

Wenn andere Ebenen aktiv sind, gelten die Fehlerbedingungen analog zu den obigen XY-Bedingungen.

Vorläufige und zwischenzeitliche Bewegung

Die vorbereitende Bewegung ist eine Gruppe von Bewegungen, die allen vorprogrammierten Fräszyklen gemeinsam ist. Wenn die aktuelle Z-Position unterhalb der R-Position liegt, führt die Z-Achse einen Eilgang zur R-Position aus. Dies geschieht nur einmal, unabhängig vom Wert von L.

Darüber hinaus werden zu Beginn des ersten Zyklus und bei jeder Wiederholung die folgenden ein oder zwei Schritte ausgeführt:

- Ein Eilgang parallel zur XY-Ebene zur angegebenen XY-Position.
- Die Z-Achse fährt im Eiltempo in die R-Position, wenn sie sich nicht bereits in der R-Position befindet.

Wenn eine andere Ebene aktiv ist, sind die vorbereitenden und die dazwischen liegenden Bewegungen analog.

Warum ein Canned Cycle (Zyklus aus der Konserve)?

Es gibt mindestens zwei Gründe für die Verwendung von Zyklen aus der Konserve. Der erste ist die Einsparung von Code. Eine einzige Bohrung würde mehrere Codezeilen benötigen, um ausgeführt zu werden.

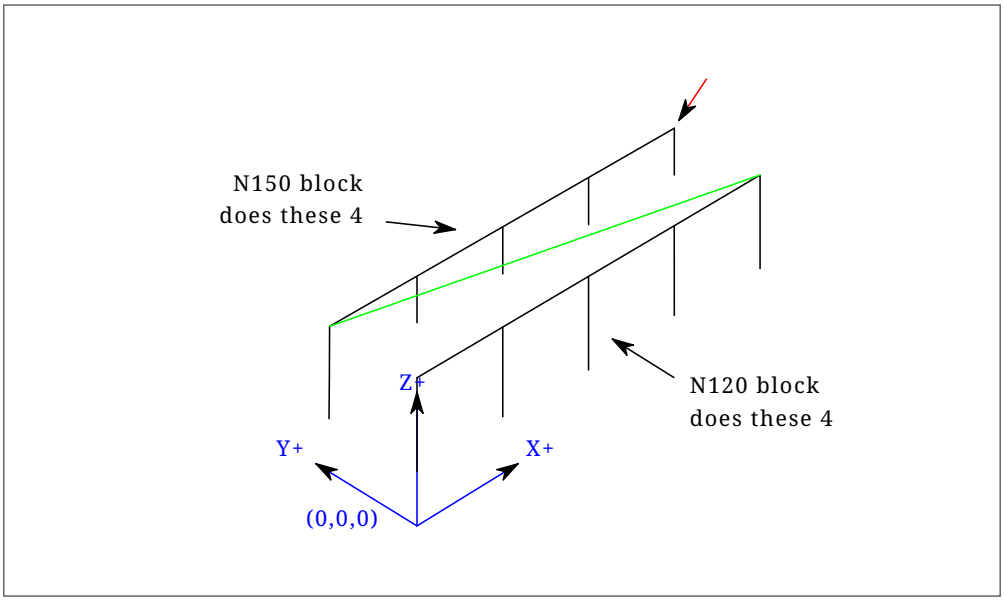
Die G81 Example 1 demonstriert, wie ein Festzyklus verwendet werden kann, um 8 Löcher mit zehn Zeilen G-Code im Festzyklusmodus zu erzeugen. Das folgende Programm erzeugt den gleichen Satz von 8 Löchern mit fünf Zeilen für den Festzyklus. Es folgt nicht genau dem gleichen Pfad und bohrt auch nicht in der gleichen Reihenfolge wie das frühere Beispiel. Aber die Wirtschaftlichkeit eines guten Festzyklus beim Programmieren sollte offensichtlich sein.

NOTE Zeilennummern sind nicht erforderlich, dienen aber der Verdeutlichung dieser Beispiele.

Acht Löcher

```
N100 G90 G0 X0 Y0 Z0 (Koordinate zum Referenzpunkt setzen)
N110 G1 F10 X0 G4 P0.1
N120 G91 G81 X1 Y0 Z-1 R1 L4 (Festbohrzyklus)
N130 G90 G0 X0 Y1
N140 Z0
N150 G91 G81 X1 Y0 Z-0,5 R1 L4(Festbohrzyklus)
N160 G80 (Ausschalten des Festzyklus)
N170 M2 (Programmende)
```

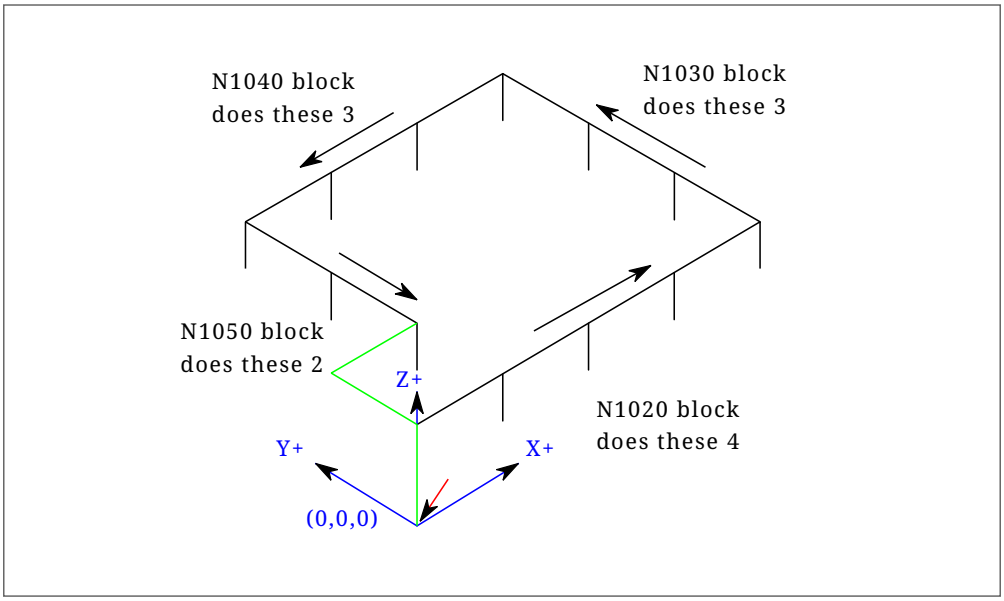
Das G98 in der zweiten Zeile oben bedeutet, dass der Rücklauf auf den Z-Wert in der ersten Zeile erfolgt, da dieser höher ist als der angegebene R-Wert.



Zwölf Löcher im Viereck

Dieses Beispiel demonstriert die Verwendung des L Werts, um eine Reihe von inkrementellen Bohrzyklen für aufeinanderfolgende Codeblöcke innerhalb desselben G81 Bewegungsmodus zu wiederholen. In diesem Beispiel werden 12 Bohrungen mit fünf Codezeilen im Modus "Festgelegte Bewegung" erzeugt.

```
N1000 G90 G0 X0 Y0 Z0 (Koordinate zum Referenzpunkt verschieben)
N1010 G1 F50 X0 G4 P0.1
N1020 G91 G81 X1 Y0 Z-0.5 R1 L4 (Festbohrzyklus)
N1030 X0 Y1 R0 L3 (Wiederholung)
N1040 X-1 Y0 L3 (Wiederholung)
N1050 X0 Y-1 L2 (Wiederholung)
N1060 G80 (Ausschalten des Festzyklus)
N1070 G90 G0 X0 (Eilgang nach Hause)
N1080 Y0
N1090 Z0
N1100 M2 (Programmende)
```



Der zweite Grund für die Verwendung eines festen Zyklus ist, dass alle diese Zyklen vorläufige Bewegungen und Erträge erzeugen, die Sie unabhängig vom Startpunkt des festen Zyklus vorhersehen und kontrollieren können.

11.5.48.G80 Festzyklus abbrechen

- *G80* - Aufhebung der modalen Bewegung des Festzyklus. *G80* ist Teil der Modalgruppe 1, so dass die Programmierung eines anderen G-Codes aus der Modalgruppe 1 auch den Festzyklus aufhebt.

Es ist ein Fehler, wenn:

- Achs-Worte werden bei aktivem G80 programmiert.

G80 Beispiel

```
G90 G81 X1 Y1 Z1.5 R2.8 (Festzyklus mit absolutem Abstand)
G80 (Ausschalten der Festzyklusbewegung)
G0 X0 Y0 Z0 (Eilgang zum Koordinatenursprung)
```

Der folgende Code ergibt dieselbe Endposition und denselben Maschinenzustand wie der vorherige Code.

G0 Beispiel

```
G90 G81 X1 Y1 Z1.5 R2.8 (absoluter Abstand im Festzyklus)
G0 X0 Y0 Z0 (Eilgang zum Koordinatenursprung)
```

Der Vorteil des ersten Satzes ist, dass die G80-Zeile den G81-Festzyklus eindeutig ausschaltet. Mit dem ersten Satz von Sätzen muss der Programmierer die Bewegung mit G0 wieder einschalten, wie es in der nächsten Zeile geschieht, oder mit einem anderen Bewegungsmodus-G-Wort.

Wenn ein Festzyklus nicht mit G80 oder einem anderen Bewegungswort ausgeschaltet wird, versucht der Festzyklus, sich mit dem nächsten Codesatz zu wiederholen, der ein X-, Y- oder Z-Wort enthält. Die folgende Datei bohrt (G81) einen Satz von acht Löchern, wie in der folgenden Beschriftung gezeigt.

G80 Beispiel 1

```
N100 G90 G0 X0 Y0 Z0 (Koordinatenheimat)
N110 G1 X0 G4 P0.1
N120 G81 X1 Y0 Z0 R1 (Festbohrzyklus)
N130 X2
N140 X3
N150 X4
N160 Y1 Z0.5
N170 X3
N180 X2
N190 X1
N200 G80 (Ausschalten des Festzyklus)
N210 G0 X0 (Eilgang nach Hause)
N220 Y0
N230 Z0
N240 M2 (Programmende)
```

NOTE	Beachten Sie die Änderung der Z-Position nach den ersten vier Löchern. Außerdem ist dies eine der wenigen Stellen, an denen Zeilennummern einen gewissen Wert haben, da sie den Leser auf eine bestimmte Codezeile verweisen können.

Die Verwendung von G80 in Zeile N200 ist optional, da das G0 in der nächsten Zeile den G81-Zyklus ausschaltet. Aber die Verwendung von G80, wie in Beispiel 1 gezeigt, macht den Festzyklus leichter lesbar. Ohne G80 ist es nicht so offensichtlich, dass alle Blöcke zwischen N120 und N200 zum Festzyklus gehören.

11.5.49.G81 Bohrzyklus

```
G81 (X- Y- Z-) oder (U- V- W-) R- L-
```

Der *G81*-Zyklus ist für Bohrungen bestimmt.

Der Zyklus funktioniert wie folgt:

- Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
- Fahren Sie die Z-Achse mit dem aktuellen Vorschub auf die Z-Position.
- Die Z-Achse führt einen Eilgang aus, um über Z den Weg freizumachen.

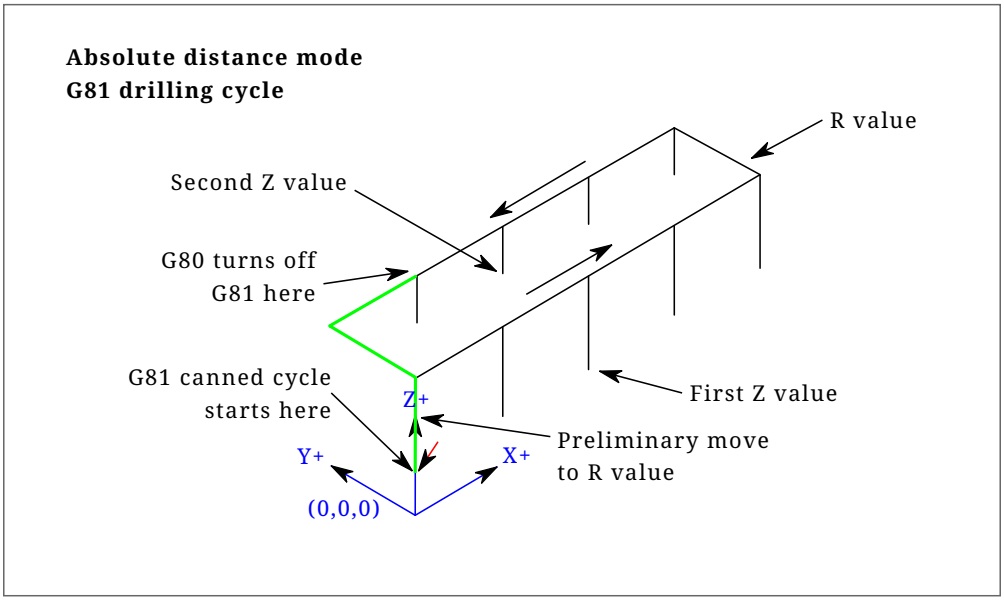


Figure 219. G81 Zyklus

Beispiel 1 – Absolute Position G81

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

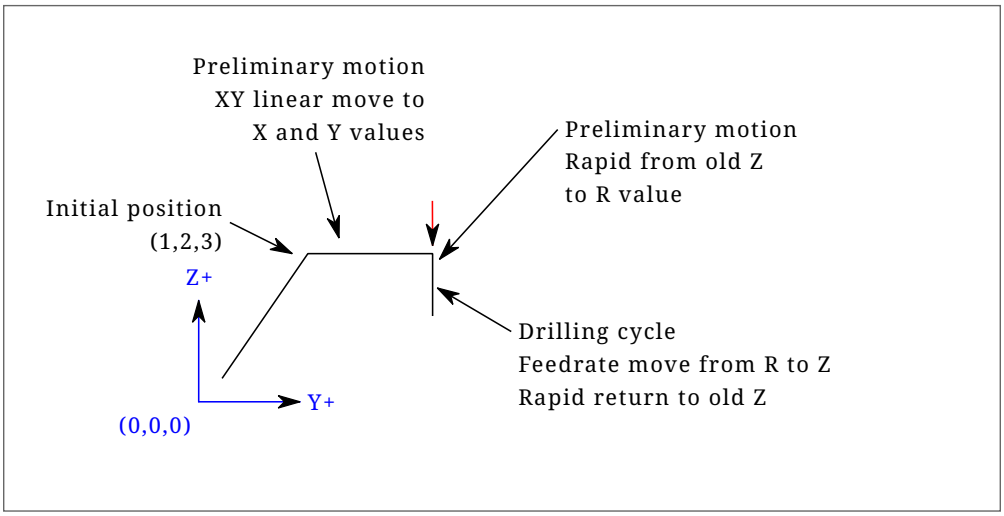
Angenommen, die aktuelle Position ist (X1, Y2, Z3) und die vorangehende Zeile des NC-Codes wird interpretiert.

Dies erfordert den absoluten Abstandsmodus (G90) und den OLD_Z-Rückzugsmodus (G98) sowie die einmalige Ausführung des Bohrzyklus G81.

- Der X-Wert und die X-Position sind 4.
- Der Y-Wert und die Y-Position sind 5.
- Der Z-Wert und die Z-Position sind 1,5.
- Der R-Wert und der klare (engl. clear) Z-Wert sind 2,8. OLD_Z ist 3.

Die folgenden Bewegungen finden statt:

- Im Eilgang parallel zur XY-Ebene nach (X4, Y5)
- Im Eilgang parallel zur Z-Achse nach (Z2.8).
- Fahrt parallel zur Z-Achse mit normalem Vorschub zu (Z1.5)
- Im Eilgang parallel zur Z-Achse nach (Z3)



Beispiel 2 – Relative Position G81

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Angenommen, die aktuelle Position ist (X1, Y2, Z3) und die vorangehende Zeile des NC-Codes wird interpretiert.

Dies erfordert den inkrementellen Abstandsmodus (G91) und den OLD_Z-Rückzugsmodus (G98). Außerdem muss der Bohrzyklus G81 dreimal wiederholt werden. Der X-Wert ist 4, der Y-Wert ist 5, der Z-Wert ist -0,6 und der R-Wert ist 1,8. Die anfängliche X-Position ist 5 (=1+4), die anfängliche Y-Position ist 7 (=2+5), die freie Z-Position ist 4,8 (=1,8+3) und die Z-Position ist 4,2 (=4,8-0,6). OLD_Z ist 3.

Die erste vorläufige Bewegung ist eine maximal schnelle Bewegung entlang der Z-Achse nach (X1,Y2,Z4.8), da OLD_Z < clear Z.

Die erste Wiederholung besteht aus 3 Zügen.

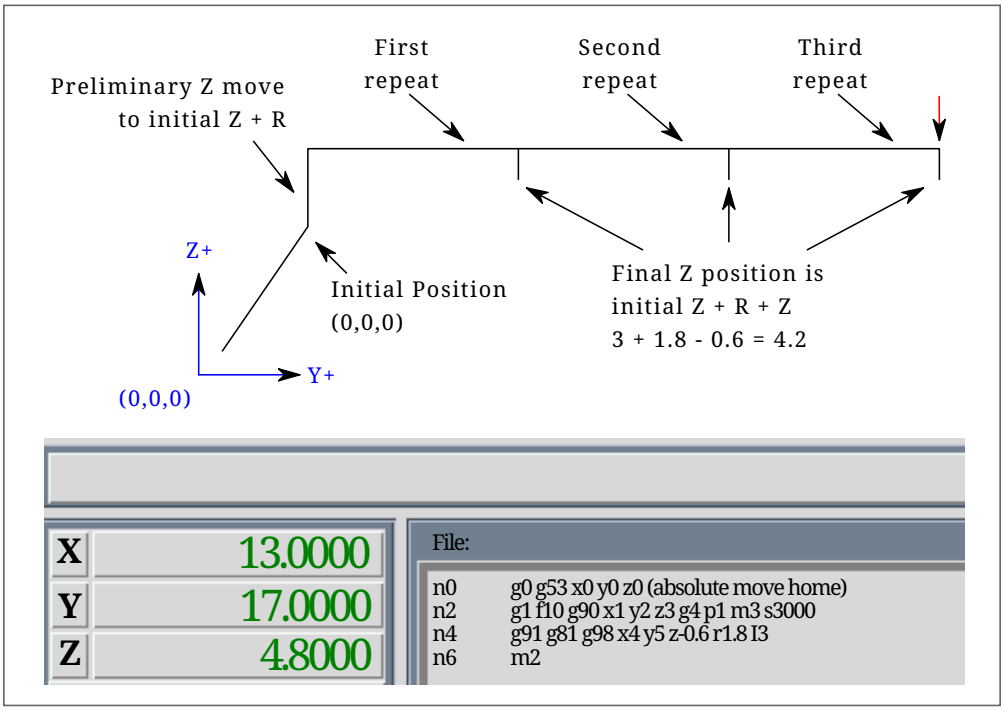
- Im Eilgang parallel zur XY-Ebene nach (X5, Y7)
- Fahrt parallel zur Z-Achse mit Vorschub-Geschwindigkeit nach (Z4.2)
- Im Eilgang parallel zur Z-Achse nach (X5, Y7, Z4.8)

Die zweite Wiederholung besteht aus 3 Zügen. Die X-Position wird auf 9 (=5+4) und die Y-Position auf 12 (=7+5) zurückgesetzt.

- Im Eilgang parallel zur XY-Ebene zu (X9, Y12, Z4.8)
- Fahrt parallel zur Z-Achse mit Vorschub-Geschwindigkeit auf (X9, Y12, Z4.2)
- Eine schnelle Bewegung parallel zur Z-Achse zu (X9, Y12, Z4.8)

Die dritte Wiederholung besteht aus 3 Zügen. Die X-Position wird auf 13 (=9+4) und die Y-Position auf 17 (=12+5) zurückgesetzt.

- Im Eilgang parallel zur XY-Ebene nach (X13, Y17, Z4.8)
- Verfahren parallel zur Z-Achse im Vorschub bis (X13, Y17, Z4.2)
- Im Eilgang parallel zur Z-Achse nach (X13, Y17, Z4.8)

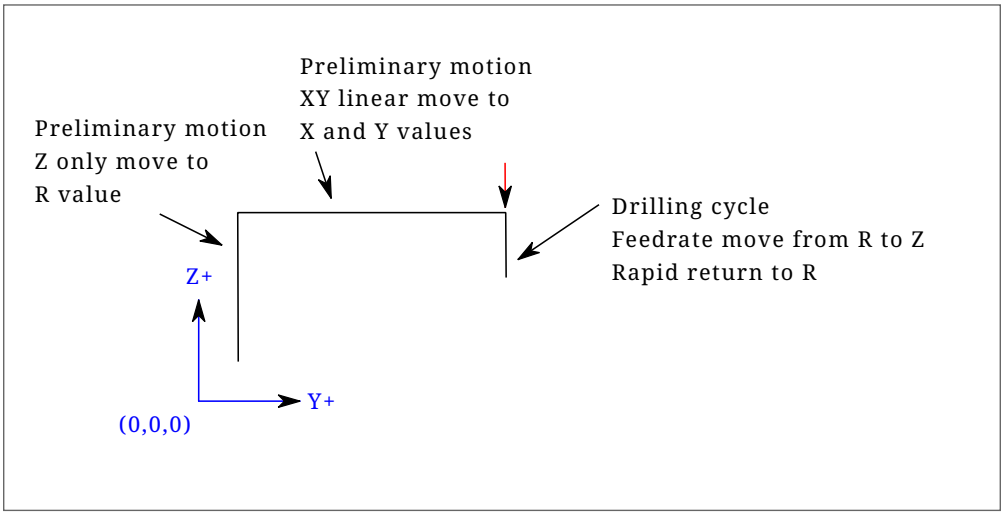


Beispiel 3 - Relative Position G81

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

Nehmen wir nun an, dass Sie den ersten G81-Codeblock ausführen, aber nicht von (X1, Y2, Z3), sondern von (X0, Y0, Z0) aus.

Da OLD_Z unter dem R-Wert liegt, fügt es der Bewegung nichts hinzu, aber da der Anfangswert von Z kleiner als der in R angegebene Wert ist, wird es während der vorbereitenden Bewegungen eine anfängliche Z-Bewegung geben.

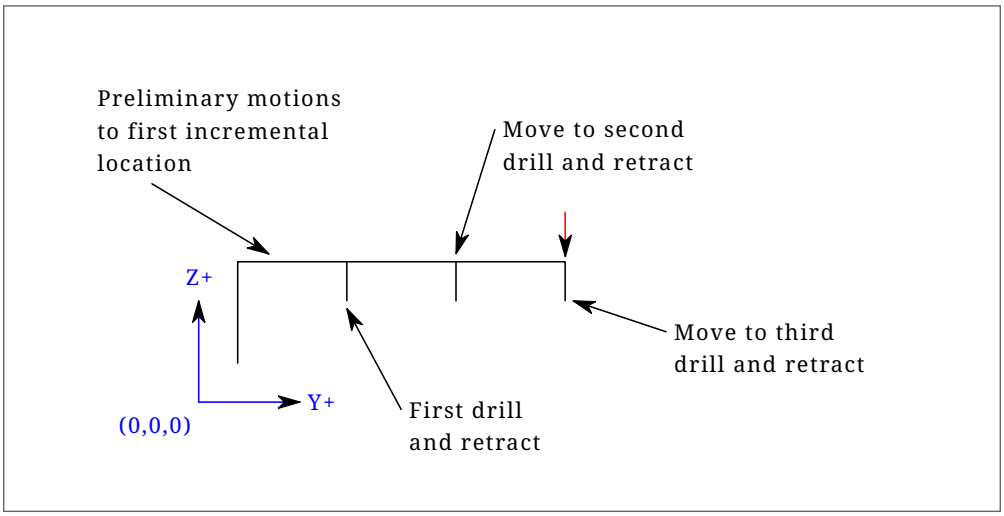


Beispiel 4 - Absolute G81 R > Z

Dies ist eine Darstellung des Bewegungspfads für den zweiten g81-Codeblock.

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Da diese Darstellung mit (X0, Y0, Z0) beginnt, fügt der Interpreter die anfängliche Z0 und R1.8 hinzu und bewegt sich schnell zu dieser Position. Nach dieser anfänglichen Z-Bewegung funktioniert die Wiederholungsfunktion genauso wie in Beispiel 3, wobei die endgültige Z-Tiefe 0,6 unter dem R-Wert liegt.



Beispiel 5 - Relative Position R > Z

```
G90 G98 G81 X4 Y5 Z-0.6 R1.8
```

Da diese Zeichnung mit (X0, Y0, Z0) beginnt, fügt der Interpreter die anfänglichen Z0 und R1.8 hinzu und bewegt sich schnell zu dieser Position wie in "Beispiel 4". Nach dieser anfänglichen Z-Bewegung wird die Eilgang-Bewegung nach X4 Y5 durchgeführt. Die endgültige Z-Tiefe liegt dann 0,6 unter dem R-Wert. Die Wiederholungsfunktion würde die Z-Bewegung an der gleichen Stelle wiederholen.

11.5.50.G82 Bohrzyklus, Verweilzeit (engl. dwell)

```
G82 (X- Y- Z-) or (U- V- W-) R- L- P-
```

Der Zyklus *G82* ist für das Bohren mit einer Verweilzeit am Bohrungsgrund vorgesehen.

- Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
- Fahren Sie die Z-Achse mit dem aktuellen Vorschub auf die Z-Position.
- Verweilen für die Anzahl von P Sekunden.
- Die Z-Achse führt einen Eilgang aus, um über Z den Weg freizumachen.

Die Bewegung eines G82-Bohrzyklus (engl. canned cycle) sieht genauso aus wie G81 mit dem Zusatz einer Verweilzeit am Ende der Z-Bewegung. Die Länge der Verweilzeit wird durch ein 'P-'Wort im G82-Satz festgelegt.

```
G90 G82 G98 X4 Y5 Z1.5 R2.8 P2
```

Dies entspricht dem obigen Beispiel 3, nur mit einer zusätzlichen Verweilzeit von 2 Sekunden am Boden des Lochs.

11.5.51.G83 Peck Bohrzyklus

```
G83 (X- Y- Z-) or (U- V- W-) R- L- Q- D-
```

The *G83* cycle (often called peck drilling) is intended for deep drilling or milling with chip breaking. The retracts in this cycle clear the hole of chips and cut off any long stringers (which are common when drilling in aluminum). This cycle takes a Q number which represents a *delta* increment along the Z-axis. The retract before final depth will always be to the *retract* plane even if G98 is in effect. The final retract will honor the G98/99 in effect. G83 functions the same as G81 with the addition of retracts during the drilling operation. Peck clearance can be specified by optional D number.

- Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
- Die Z-Achse mit dem aktuellen Vorschub um Delta nach unten oder auf die Z-Position fahren, je nachdem, was weniger tief ist.
- Schnelles Zurückfahren auf die durch das R-Wort angegebene Rückzugsebene.
- Rapid up either the D value, the G83_PECK_CLEARANCE specified in the INI file or the default of .010" / 0.254 mm.
- Wiederholen der Schritte 2, 3 und 4, bis die Z-Position bei Schritt 2 erreicht ist.
- Die Z-Achse führt einen Eilgang aus, um über Z den Weg freizumachen.

Es ist ein Fehler, wenn:

- Die Q-Zahl ist negativ oder null.

11.5.52.G84 Rechter Gewindeschneidzyklus, Verweilzeit

```
G84 (X- Y- Z-) or (U- V- W-) R- L- P- $- F-
```

- *R-* - Zurückziehen der Position entlang der Z-Achse.
- L-' - Wird im inkrementellen Modus verwendet; Anzahl der Wiederholungen des Zyklus. Siehe G81 für Beispiele.
- P-' - Verweilzeit (Sekunden).
- *$-* - Ausgewählte Spindel.
- *F-* - Vorschubgeschwindigkeit (Spindeldrehzahl multipliziert mit der pro Umdrehung zurückgelegten Strecke (Gewindesteigung)).

WARNING	G84 verwendet keine synchronisierte Bewegung.

Der Zyklus *G84* ist für das Gewindeschneiden mit schwimmendem Spannfutter und Verweilzeit am Bohrungsgrund vorgesehen.

- Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
- Deaktivieren von Vorschub- und Geschwindigkeits-Neufestsetzungen (engl. overrides).
- Fahren Sie die Z-Achse mit der aktuellen Vorschubgeschwindigkeit in die Z-Position.
- Anhalten der ausgewählten Spindel (ausgewählt durch den Parameter $)
- Starten Sie die Spindeldrehung gegen den Uhrzeigersinn.
- Verweilen für die Anzahl von P Sekunden.
- Bewegen Sie der Z-Achse mit der aktuellen Vorschubgeschwindigkeit, um Z zu löschen
- Wiederherstellung der Vorschub- und Geschwindigkeitsneufestsetzung-Aktivierung in den vorherigen Zustand

Die Länge der Verweilzeit wird durch ein *P*--Wort im G84-Satz angegeben. Die Vorschubgeschwindigkeit *F-* ist die Spindeldrehzahl multipliziert mit dem Abstand pro Umdrehung (Gewindesteigung). Im Beispiel S100 mit 1,25MM pro Umdrehung Gewindesteigung ergibt einen Vorschub von F125.

11.5.53.G85 Bohrzyklus, Verweilen, Rückzug in Vorschubgeschwindigkeit

```
G85 (X- Y- Z-) or (U- V- W-) R- L-
```

Der Zyklus "G85" ist zum Bohren oder Reiben vorgesehen, kann aber auch zum Bohren oder Fräsen verwendet werden.

- Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
- Bewege die Z-Achse nur mit dem aktuellen Vorschub zur Z-Position.
- Rückzug der Z-Achse mit dem aktuellen Vorschub in die R-Ebene, wenn dieser niedriger ist als der ursprüngliche Z-Wert.

- Mit der Verfahrgeschwindigkeit einfahren, um Z zu löschen.

11.5.54.G86 Bohrzyklus, Spindel Halt, Eilgang raus

```
G86 (X- Y- Z-) oder (U- V- W-) R- L- P- $-
```

Der Zyklus "G86" ist für das Bohren vorgesehen. Dieser Zyklus verwendet eine P-Zahl für die Anzahl der Verweilsekunden.

- Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
- Bewege die Z-Achse nur mit dem aktuellen Vorschub zur Z-Position.
- Verweilen für die Anzahl von P Sekunden.
- Stoppt die Drehung der ausgewählten Spindel. (Wird durch den Parameter $ ausgewählt)
- Die Z-Achse führt einen Eilgang aus, um über Z den Weg freizumachen.
- Starten Sie die Spindel wieder in der ursprünglichen Richtung.

Es ist ein Fehler, wenn:

- die Spindel sich nicht dreht, bevor dieser Zyklus ausgeführt wird.

11.5.55.G87 Rückbohren des Zyklus

Dieser Code ist derzeit nicht in LinuxCNC implementiert. Es wird akzeptiert, aber das Verhalten ist undefiniert.

11.5.56.G88 Bohrzyklus, Spindel Halt, manuelles Herausziehen

Dieser Code ist derzeit nicht in LinuxCNC implementiert. Es wird akzeptiert, aber das Verhalten ist undefiniert.

11.5.57.G89 Bohrzyklus, Verweilen, Vorschub Raus

```
G89 (X- Y- Z-) or (U- V- W-) R- L- P-
```

Der Zyklus *G89* ist für das Bohren vorgesehen. Dieser Zyklus verwendet eine P-Zahl, wobei P die Anzahl der zu verweilenden Sekunden angibt.

- Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
- Bewege die Z-Achse nur mit dem aktuellen Vorschub zur Z-Position.
- Verweilen für die Anzahl von P Sekunden.
- Zurückfahren der Z-Achse mit der aktuellen Vorschubgeschwindigkeit, um Z zu löschen.

11.5.58.G90, G91 Distanz-Modus

- *G90* - absoluter Abstandsmodus Im absoluten Abstandsmodus stellen die Achsennummern (X, Y, Z, A, B, C, U, V, W) normalerweise Positionen in Bezug auf das gerade aktive Koordinatensystem dar. Alle Ausnahmen von dieser Regel werden im Abschnitt G80 G89 ausdrücklich beschrieben.
- "G91' - inkrementeller Abstandsmodus. Im inkrementellen Abstandsmodus stellen die Achsennummern normalerweise Zuwächse (engl. increments) ausgehend von der aktuellen Koordinate dar.

G90 Beispiel

```
G90 (Einstellung des absoluten Abstandsmodus)
G0 X2.5 (Schnellverschiebung zur Koordinate X2.5, einschließlich eventueller Offsets)
```

G91 Beispiel

```
G91 (Inkrementellen Abstandsmodus einstellen)
G0 X2.5 (Eilgang 2,5 von der aktuellen Position entlang der X-Achse)
```

- Siehe Abschnitt G0 für weitere Informationen.

11.5.59.G90.1, G91.1 Bogenabstands-Modus (engl. Arc Distance Mode)

- *G90.1* - absoluter Abstandsmodus für I, J und K Versatz. Wenn G90.1 in Kraft ist, müssen I und J beide mit G2/3 für die XY-Ebene oder J und K für die XZ-Ebene angegeben werden, sonst ist es ein Fehler.
- *G91.1* - inkrementeller Abstandsmodus für I, J und K Offsets. `G91.1` Setzt I, J und K auf ihren Standardwert zurück.

11.5.60.G92 Koordinatensystem-Versatz

```
G92 Achsen
```

WARNING Verwenden Sie *G92* erst, nachdem Ihre Maschine am gewünschten Punkt positioniert wurde.

G92 bewirkt, dass der aktuelle Punkt die gewünschten Koordinaten hat (ohne Bewegung), wobei die Achsenwörter die gewünschten Achsennummern enthalten. Alle Achswörter sind optional, außer dass mindestens eines verwendet werden muss. Wenn für eine bestimmte Achse kein Achsenwort verwendet wird, ist der Offset für diese Achse null.

Wenn *G92* ausgeführt wird, verschieben sich die Ursrpünge (engl. origins) aller Koordinatensysteme. Sie verschieben sich so, dass der Wert des aktuell kontrollierten Punktes im aktuell aktiven Koordinatensystem den angegebenen Wert annimmt. Alle Ursprünge des Koordinatensystems (G53-G59.3) werden um denselben Abstand verschoben.

G92' verwendet die in parameters 5211-5219 gespeicherten Werte als X Y Z A B C U V W Offsetwerte für jede Achse. Die Parameterwerte sind "absolute" Maschinenkoordinaten in den nativen Maschineneinheiten, wie in der INI-Datei angegeben. Alle in der INI-Datei definierten Achsen werden

versetzt, wenn G92 aktiv ist. Wenn eine Achse nach G92 nicht eingegeben wurde, ist der Offset dieser Achse Null.

Angenommen, der aktuelle Punkt befindet sich bei X=4 und es ist derzeit kein *G92*-Offset aktiv. Dann wird *G92 X7* programmiert. Dadurch werden alle Ursprünge um -3 in X verschoben, wodurch der aktuelle Punkt zu X=7 wird. Diese -3 wird in Parameter 5211 gespeichert.

Der inkrementelle Entfernungsmodus (G91 statt G90) hat keinen Einfluss auf die Wirkung von G92.

Die *G92*-Versätze können bereits in Kraft sein, wenn G92 aufgerufen wird. Ist dies der Fall, wird der Offset durch einen neuen Offset ersetzt, der den aktuellen Punkt zu dem angegebenen Wert macht.

Es ist ein Fehler, wenn alle Achsworte weggelassen werden.

LinuxCNC speichert die G92 Offsets und wiederverwendet sie auf den nächsten Lauf eines Programms. Um dies zu verhindern, kann man ein G92.1 programmieren (um sie zu löschen), oder ein G92.2 programmieren (um sie zu entfernen - sie sind immer noch gespeichert).

NOTE Der Befehl *G52* kann auch verwendet werden, um diesen Offset zu ändern; siehe den Abschnitt Lokaler und Globaler Versatz für weitere Einzelheiten über G92 und G52 und wie sie zusammenwirken.

Einen Überblick über Koordinatensysteme finden Sie im Abschnitt Koordinatensystem.

Siehe den Abschnitt zu Parametern für weitere Informationen.

11.5.61.G92.1, G92.2 Zurücksetzen der G92 Offsets

- *G92.1* - Schalten Sie G92-Offsets aus und setzt Parameter 5211 - 5219 auf Null zurück.
- *G92.2* - G92-Offsets ausschalten, aber Parameter 5211 - 5219 verfügbar lassen.

NOTE G92.1 löscht nur G92-Offsets, um G53-G59.3-Koordinatensystem-Offsets im G-Code zu ändern, verwenden Sie entweder G10 L2 oder G10 L20.

11.5.62.G92.3 Wiederherstellen des G92 Versatzes

- G92.3 - setzt den G92-Offset auf die in den Parametern 5211 bis 5219 gespeicherten Werte

Sie können Achsversätze in einem Programm einstellen und die gleichen Versätze in einem anderen Programm verwenden. Programmieren Sie G92 im ersten Programm. Dadurch werden die Parameter 5211 bis 5219 eingestellt. Verwenden Sie im weiteren Verlauf des ersten Programms nicht G92.1. Die Parameterwerte werden beim Verlassen des ersten Programms gespeichert und beim Starten des zweiten Programms wiederhergestellt. Verwenden Sie G92.3 zu Beginn des zweiten Programms. Dadurch werden die im ersten Programm gespeicherten Offsets wiederhergestellt.

11.5.63.G93, G94, G95 Vorschubgeschwindigkeits-Modus(engl. feed rate modes)

- *G93* - ist der Modus "Inverse Zeit". Im Modus für inverse Zeitvorschübe bedeutet ein F-Wert, dass die Bewegung in [eins geteilt durch die F-Zahl] Minuten abgeschlossen sein sollte. Wenn die F-Zahl beispielsweise 2.0 beträgt, sollte die Bewegung in einer halben Minute abgeschlossen sein.

  Wenn der Modus für den inversen Zeitvorschub aktiv ist, muss ein F-Wort in jeder Zeile erscheinen, die eine G1-, G2- oder G3-Bewegung hat, und ein F-Wort in einer Zeile, die keine `G1`-, `G2`- oder `G3` -Bewegung hat, wird ignoriert. Der Modus "Inverser Zeitvorschub" wirkt sich nicht auf `G0` -Bewegungen (rapid move) aus.

- "G94" - ist der Modus Einheiten pro Minute. Im Vorschubmodus Einheiten pro Minute wird ein F-Wert so interpretiert, dass der gesteuerte Punkt mit einer bestimmten Anzahl von Zoll pro Minute, Millimeter pro Minute oder Grad pro Minute verfahren werden soll, je nachdem, welche Längeneinheiten verwendet werden und welche Achse oder Achsen sich bewegen.
- *G95* - ist der Einheiten-pro-Umdrehungs-Modus Im Einheiten-pro-Umdrehungs-Modus wird ein F-Wort dahingehend interpretiert, dass sich der gesteuerte Punkt um eine bestimmte Anzahl von Zoll pro Spindelumdrehung bewegen soll, je nachdem, welche Längeneinheiten verwendet werden und welche Achse oder Achsen bewegen sich. G95 ist nicht zum Gewindeschneiden geeignet, zum Gewindeschneiden `G33` oder `G76` verwenden. `G95` erfordert, dass `spindle.`*N*`.speed-in` verbunden ist. Durch den $-Parameter wird die tatsächliche Spindel ausgewählt, auf die der Vorschub synchronisiert wird.

Es ist ein Fehler, wenn:

- Der inverse Zeitvorschubmodus ist aktiv und eine Zeile mit `G1`, `G2` oder `G3` (explizit oder implizit) hat kein F-Wert.
- Nach dem Umschalten auf `G94` oder `G95` keine neue Vorschubgeschwindigkeit angegeben wird.

11.5.64.G96, G97 Spindelsteuerungs-Modus

```
G96 <D-> S- <$-> (Modus konstante Oberflächengeschwindigkeit)
G97 S- <$-> (Drehzahlmodus)
```

1. *D* - maximum rotation speed (RPM), optional. Without D the spindle speed in CSS mode is not limited by the interpreter.
2. *S* - Spindeldrehzahl
3. $' - die Spindel, deren Geschwindigkeit variiert werden soll, optional.
   - G96 S- <D→' - wählt eine konstante Oberflächengeschwindigkeit von *S*:
     - In Fuß pro Minute, wenn `G20` aktiviert ist,
     - oder Meter pro Minute, wenn `G21` aktiv ist.

Stellen Sie bei Verwendung von G96 sicher, dass X0 im aktuellen Koordinatensystem (einschließlich Offsets und Werkzeuglängen) das Rotationszentrum ist, da LinuxCNC sonst nicht die gewünschte Oberflächengeschwindigkeit liefert. G96 wird vom Radius- oder Durchmessermodus nicht beeinflusst.

Um den CSS-Modus für ausgewählte Spindeln zu erreichen, programmieren Sie vor der Ausgabe von M3 aufeinanderfolgende G96-Befehle für jede Spindel.

- *G97* wählt den Drehzahlmodus aus.

G96 Beispielzeile

```
G96 D2500 S250 (CSS mit einer maximalen Drehzahl von 2500 und einer
Oberflächengeschwindigkeit von 250)
```

Es ist ein Fehler, wenn:

- S ist bei G96 nicht festgelegt

11.5.65.G98, G99 Festzyklus-Rückkehrniveau

Wenn die Spindel während der Festzyklen zurückfährt, gibt es zwei Möglichkeiten, wie dies geschehen kann:

- *G98* - Rückzug auf die Position, in der sich die Achse befand, kurz bevor diese Serie von einem oder mehreren zusammenhängenden Festzyklen gestartet wurde.
- *G99* - Rückzug bis zu der durch den R-Wert festgelegten Position des Festzyklus.

Programmieren Sie ein *G98* Kommando und der Festzyklus verwendet die Z-Position vor dem Festzyklus als Z-Rückkehrposition, wenn sie höher ist als der im Zyklus angegebene R-Wert. Wenn sie niedriger ist, wird der R-Wert verwendet. Das R-Wort hat im absoluten Abstandsmodus und im inkrementellen Abstandsmodus unterschiedliche Bedeutungen.

G98 Zurückfahren zum Ausgangspunkt

```
G0 X1 Y2 Z3
G90 G98 G81 X4 Y5 Z-0.6 R1.8 F10
```

Das G98 in der zweiten Zeile im obigen Beispiel bedeutet, dass der Rücksprung auf den Wert von Z in der ersten Zeile erfolgt, da dieser höher ist als der angegebene R-Wert.

Die *Ausgangsebene* (G98) wird jedes Mal zurückgesetzt, wenn der Zyklusmodus verlassen wird, sei es explizit (G80) oder implizit (jeder Bewegungscode, der kein Zyklus ist). Ein Wechsel zwischen den Zyklusmodi (z. B. G81 zu G83) setzt die "Ausgangsebene" NICHT zurück. Es ist möglich, während einer Reihe von Zyklen zwischen G98 und G99 zu wechseln.

11.6. M-Codes

11.6.1. M-Code Kurzübersichts-Ttabelle

Code	Beschreibung
M0 M1	Programm-Pause

Code	Beschreibung
M2 M30	Programmende
M60	Palettenwechsel Pause
M3 M4 M5	Spindelsteuerung
M6	Werkzeugwechsel
M7 M8 M9	Kühlmittelkontrolle
M19	Spindel ausrichten
M26 M27	Rotary Modulo Path: Shortest / Literal
M48 M49	Vorschub- und Spindel-Neufestsetzungen (engl. overrides) aktivieren/deaktivieren
M50	Kontrolle der Vorschub-Neufestsetzung (engl. override)
M51	Kontrolle der Spindel-Neufestsetzung (engl. Override)
M52	Adaptive Vorschubsteuerung
M53	Steuerung des Vorschub-Stopps
M61	Aktuelle Werkzeugnummer einstellen
m62-m65	Ausgangs-Steuerung (engl. output control)
M66	Eingabe-Steuerung (engl. input control)
M67	Steuerung des Analogausgangs
M68	Steuerung des Analogausgangs
M70	Modalstatus speichern
M71	Gespeicherten Modalzustand ungültig machen

Code	Beschreibung
M72	Modalen Zustand wiederherstellen
M73	Modalzustand der automatischen Wiederherstellung (engl. autorestore) speichern
M98 M99	Aufruf und Rückkehr aus Unterprogramm
M100-M199	Anwender-definierte M-Codes

11.6.2. M0, M1 Programm-Pause

- M0' - pausiert ein laufendes Programm vorübergehend. LinuxCNC bleibt im Auto-Modus, somit sind MDI und andere manuelle Aktionen sind nicht aktiviert. Durch Drücken der Taste *Fortführen* (engl. *Resume*) wird das Programm in der folgenden Zeile weiterlaufen.
- M1' - pausiert ein laufendes Programm vorübergehend, wenn der optionale Stop-Schalter eingeschaltet ist. LinuxCNC bleibt im Auto-Modus so MDI und andere manuelle Aktionen sind nicht aktiviert. Durch Drücken der Resume-Taste wird das Programm in der folgenden Zeile neu gestartet.

NOTE Es ist in Ordnung, *M0* und *M1* im MDI-Modus zu programmieren, aber der Effekt wird wahrscheinlich nicht spürbar sein, weil das normale Verhalten im MDI-Modus ohnehin darin besteht, nach jeder Eingabezeile anzuhalten.

11.6.3. M2, M30 Programmende

- *M2* - Programm beenden. Durch Drücken von *Zyklusstart* (engl. Cycle Start, *R* im Axis-GUI) wird das Programm am Anfang der Datei neu gestartet.
- *M30* - tauschen Sie Paletten-Shuttles aus und beenden Sie das Programm. Wenn Sie auf `Cycle Start` klicken, wird das Programm am Anfang der Datei gestartet.

Diese beiden Befehle haben die folgenden Auswirkungen:

- Wechsel vom Auto-Modus in den MDI-Modus.
- Origin offsets are set to the default (like *G54*) unless disabled by INI setting *DISABLE_AUTO_G54 = 1*.
- A tilted work plane is cancelled (like *G69*) unless kept by INI setting *RETAIN_WORK_PLANE = 1*.
- Die ausgewählte Ebene ist auf die XY-Ebene eingestellt (wie *G17*).
- Der Abstandsmodus ist auf den absoluten Modus eingestellt (wie *G90*).

- Der Vorschubmodus ist auf Einheiten pro Minute eingestellt (wie *G94*).
- Vorschub- und Geschwindigkeits-Neufestsetzungen (engl. overrides) sind auf ON gesetzt (wie *M48*).
- Fräserkompensation ist ausgeschaltet (wie *G40*).
- Die Spindel wird angehalten (wie bei *M5*).
- Der aktuelle Bewegungsmodus ist auf Vorschub eingestellt (wie *G1*).
- Das Kühlmittel ist ausgeschaltet (wie *M9*).
- Rotary modulo path control is set to shortest path (like *M26*).

NOTE Codezeilen nach M2/M30 werden nicht ausgeführt. Durch Drücken von *Zyklusstart* wird das Programm am Anfang der Datei gestartet.

WARNING Die Verwendung von % zum Einschließen des G-Codes bewirkt nicht dasselbe wie ein "Programmende". Siehe den Abschnitt zu Datei-Anforderungen (engl. file requirements) für weitere Informationen darüber, was die Verwendung von % nicht bewirkt.

11.6.4. M60 Palettenwechsel Pause

- *M60* - Palettenshuttle tauschen und dann ein laufendes Programm vorübergehend pausieren (unabhängig von der Einstellung des optionalen Stoppschalters). Durch Drücken der Zyklusstarttaste wird das Programm in der folgenden Zeile neu gestartet.

11.6.5. M3, M4, M5 Spindelsteuerung

- *M3 [$n]* - startet die ausgewählte Spindel im Uhrzeigersinn mit der Geschwindigkeit *S*.
- *M4 [$n]* - startet die ausgewählte Spindel gegen den Uhrzeigersinn mit der Geschwindigkeit *S*.
- *M5 [$n]* - stoppt die ausgewählte Spindel.

Verwenden Sie $, um auf bestimmte Spindeln zu wirken. Wenn $ weggelassen wird, arbeiten die Befehle standardmäßig an der Spindel 0. Verwenden Sie $-1, um an allen aktiven Spindeln zu arbeiten.

In diesem Beispiel werden die Spindeln 0, 1 und 2 gleichzeitig mit unterschiedlichen Drehzahlen gestartet:

```
S100 $0
S200 $1
S300 $2
M3 $-1
```

In diesem Beispiel wird die Spindel 1 umgekehrt, während die anderen Spindeln vorwärts laufen:

```
M4 $1
```

Dadurch wird Spindel 2 angehalten und die anderen Spindeln drehen sich weiter:

```
M5 $2
```

Wenn das $ weggelassen wird, ist das Verhalten genau wie bei einer Einspindelmaschine.

Es ist OK, *M3* oder *M4* zu verwenden, wenn die S Spindeldrehzahl auf Null gesetzt ist. In diesem Fall (oder wenn der Drehzahl-Neufestsetzungs (engl. override)-Schalter aktiviert und auf Null gesetzt ist), beginnt die Spindel nicht zu drehen. Wenn die Spindeldrehzahl später auf einen Wert über Null gesetzt wird (oder der Override-Schalter aktiviert wird), beginnt sich die Spindel zu drehen. Es ist in Ordnung, *M3* oder *M4* zu verwenden, wenn sich die Spindel bereits dreht, oder *M5* zu verwenden, wenn die Spindel bereits angehalten ist.

11.6.6. M6 Werkzeugwechsel

Manueller Werkzeugwechsel

Wenn die HAL-Komponente *hal_manualtoolchange* geladen ist, hält M6 die Spindel an und fordert den Benutzer auf, das Werkzeug auf der Grundlage der zuletzt programmierten 'T-'Nummer zu wechseln. Für weitere Informationen zu hal_manualtoolchange siehe den Abschnitt Manuelle Werkzeugwechsel.

Werkzeugwechsler

Um ein Werkzeug in der Spindel von dem Werkzeug, das sich gerade in der Spindel befindet, auf das zuletzt ausgewählte Werkzeug zu wechseln (mit einem T-Wort - siehe Abschnitt Werkzeug-Auswahl) (engl. Select Tool), programmieren Sie *M6*. Wenn der Werkzeugwechsel abgeschlossen ist:

- Die Spindel wird angehalten.
- Das Werkzeug, das (durch ein T-Wort in derselben Zeile oder in einer beliebigen Zeile nach dem letzten Werkzeugwechsel) ausgewählt wurde, befindet sich in der Spindel.
- Wenn sich das ausgewählte Werkzeug vor dem Werkzeugwechsel nicht in der Spindel befand, wird das Werkzeug, das sich in der Spindel befand (falls vorhanden), wieder in das Magazin des Werkzeugwechslers eingesetzt.
- Wenn in der INI-Datei konfiguriert, können sich einige Achsenpositionen bewegen, wenn ein M6 ausgegeben wird. Siehe EMCIO Abschnitt für weitere Informationen über Werkzeugwechseloptionen.
- Es werden keine weiteren Änderungen vorgenommen. Beispielsweise fließt während des Werkzeugwechsels weiterhin Kühlmittel, es sei denn, es wurde durch ein *M9* abgeschaltet.

NOTE	Das *T*--Wort ist eine ganze Zahl, welche die Werkzeugfachnummer (engl. tool pocket number) im Karussell bezeichnet (nicht seinen Index).

WARNING	Der Werkzeuglängen-Offset wird durch *M6* nicht verändert, verwenden Sie *G43*

nach dem *M6*, um den Werkzeuglängen-Offset zu ändern.

Der Werkzeugwechsel kann eine Achsbewegung beinhalten. Es ist in Ordnung (aber nicht sinnvoll), einen Wechsel des Werkzeugs zu programmieren, das sich bereits in der Spindel befindet. Es ist in Ordnung, wenn sich kein Werkzeug auf dem gewählten Steckplatz befindet; in diesem Fall ist die Spindel nach dem Werkzeugwechsel leer. Wenn zuletzt der Steckplatz Null gewählt wurde, befindet sich nach einem Werkzeugwechsel definitiv kein Werkzeug in der Spindel. Der Werkzeugwechsler muss so eingestellt werden, dass er den Werkzeugwechsel in HAL und bevorzugt in ClassicLadder durchführt.

11.6.7. M7, M8, M9 Kühlmittelsteuerung

- *M7* - Kühlmittelnebel (engl. mist coolant) einschalten. M7 steuert den iocontrol.0.coolant-mist Pin.
- *M8* - Kühlmittelflutung (engl. flood cloolant) einschalten. M8 steuert iocontrol.0.coolant-flood Pin.
- *M9* - sowohl M7 als auch M8 ausschalten.

Verbindet einen oder beide Kühlmittelkontrollstifte in HAL, bevor M7 oder M8 einen Ausgang steuern. M7 und M8 können verwendet werden, um einen beliebigen Ausgang über G-Code einzuschalten.

Sie können jeden dieser Befehle verwenden, unabhängig vom aktuellen Kühlmittelzustand.

11.6.8. M19 Spindel ausrichten

```
M19 R- Q- [P-] [$-]
```

- *R* Position zum Drehen von 0, gültiger Bereich ist 0-360 Grad
- *Q* Anzahl der Sekunden, die gewartet werden müssen, bis die Ausrichtung abgeschlossen ist. Wenn spindle.N.is-oriented innerhalb des Q-Timeouts nicht wahr wird, tritt ein Fehler auf.
- *P* Drehrichtung zur Position.
  - *0* drehen für kleinste Winkelbewegung (Standard)
  - *1* immer im Uhrzeigersinn drehen (wie M3-Richtung)
  - *2* immer gegen den Uhrzeigersinn drehen (wie M4-Richtung)
  - *3* same as *0* but with preceding rotation to an index signal
  - *4* same as *1* but with preceding rotation to an index signal
  - *5* same as *2* but with preceding rotation to an index signal
- *$* The spindle to orient (actually only determines which HAL pins carry the spindle position commands)

M19 ist ein Befehl der Modalgruppe 7, wie M3, M4 und M5. M19 wird durch einen der Befehle M3, M4, M5 gelöscht.

Spindle orientation with modes 0,1,2 requires a quadrature encoder to sense the spindle shaft position and direction of rotation. Spindle orientation with modes 3,4,5 requires an additional index signal for referencing the spindle before orientation.

INI-Einstellungen im Abschnitt [RS274NGC]:

- ORIENT_OFFSET = 0-360 (fester Offset in Grad, der zum M19 R-Wort hinzugefügt wird)
- HAL-Pins
  - *spindle.N.orient-angle* (real, out) Desired spindle orientation for M19. Value of the M19 R word parameter plus the value of the [RS274NGC]ORIENT_OFFSET INI parameter.
  - *spindle.N.orient-mode* (sint, out) Desired spindle rotation mode. Reflects M19 P parameter word, default = 0.
  - *spindle.N.orient* (bool, out) Indicates start of spindle orient cycle. Set by M19. Cleared by any of M3,M4,M5. If spindle-orient-fault is not zero during spindle-orient true, the M19 command fails with an error message.
  - *spindle.N.is-oriented* (bool, in) Acknowledge pin for spindle-orient. Completes orient cycle. If spindle-orient was true when spindle-is-oriented was asserted, the spindle-orient pin is cleared and the spindle-locked pin is asserted. Also, the spindle-brake pin is asserted.
  - *spindle.N.orient-fault* (sint, in) Fault code input for orient cycle. Any value other than zero will cause the orient cycle to abort.
  - *spindle.N.locked* (bool, out) Spindle orient complete pin. Cleared by any of M3,M4,M5.

11.6.9. M26, M27 Rotary Modulo Path Control

- *M26* - on rotary axes flagged with `[AXIS_<L>] ROTARY_MODULO = 1`, absolute moves take the shortest path: a programmed value is reached by moving at most 180 degrees in the shorter direction. A move of exactly 180 degrees is taken in the negative direction. A value that wraps to the angle the axis is already standing on commands no motion: `G0 A360` from `A0` leaves the axis where it is. This is the default when `ROTARY_MODULO` is enabled.
- *M27* - on the same axes, absolute moves take the modulo-with-sign-direction path: the programmed value is reduced mod 360, and the sign of the input picks the direction of approach. A positive (or zero) value approaches in the forward direction; a negative value approaches in the reverse direction. The resulting move is always less than one full turn. Programmed values outside +/-360 are accepted and wrapped, so `A720` is treated as `A0`, `A-540` as `A-180`. An axis already standing on the programmed angle does not move, whatever the sign: `G0 A360` and `G0 A-360` from `A0` both leave the axis where it is. And the commanded travel changes abruptly right at the wrap point: from `A0`, `A359.999` moves 359.999 degrees forward and `A-359.999` moves only 0.001 degrees backward - both land on the same angle, 359.999. Use this when sign convention must select rotation direction explicitly, similar to a wrapped rotary axis or to Heidenhain M127 / Siemens ACP/ACN behavior on a modulo axis.

M26 and M27 are modal (modal group 3) and persist until the other code is programmed or the program ends. They have no effect on rotary axes that are not flagged `ROTARY_MODULO`, on incremental moves

(G91), or on parameter readout (#5423/#5424/#5425 continue to report the wrapped value). Moves to a stored position that carries no programmed sign (G28, G30, the tool change position) always take the shortest path, M27 or not.

Typical usage:

```
(at this point absolute A moves take the shortest path)
M27         (enter sign-direction mode for the section that follows)
G1 A45      (forward to 45 degrees within the current turn)
G1 A-90     (backward to 90 degrees, taking the long way round)
M26         (restore shortest path for the rest of the program)
G0 A0       (now resolves to the nearest equivalent angle)
```

For multi-turn motion (helical winding, wire wrapping, etc.), use G91 incremental mode. G91 A7200 performs twenty turns regardless of ROTARY_MODULO or M26/M27 state, and matches the recommended idiom on Heidenhain (IA+) and Siemens controls.

M26 and M27 are saved and restored by M70/M72/M73 alongside the other modal M-codes.

See also: ROTARY_MODULO in [AXIS_<L>] Section.

11.6.10. M48, M49 Geschwindigkeits- und Vorschub-Override-Steuerung

- *M48* - aktiviert die Neufestsetzungs-Steuerungen für Spindeldrehzahl und Vorschubgeschwindigkeit.
- *M49* - beide Steuerelemente deaktivieren.

Diese Befehle benötigen auch einen optionalen $-Parameter, um festzulegen, auf welcher Spindel sie arbeiten.

Es ist OK, die Kontrollen zu aktivieren oder zu deaktivieren, wenn sie bereits aktiviert oder deaktiviert sind. Siehe den Vorschubgeschwindigkeit Abschnitt für weitere Details.

Sie können auch einzeln mit *M50* und *M51* umgeschaltet werden, siehe unten.

11.6.11. M50 Steuerung der Vorschub-Neufestsetzung (engl. override)

- *M50 <P1>* - aktiviert die Vorschub-Override-Steuerung. Der P1 ist optional.
- *M50 P0* - Vorschubsteuerung deaktivieren.

Im deaktivierten Zustand hat der Vorschub-Neufestsetzung (engl. override) keinen Einfluss, und die Bewegung wird mit der programmierten Vorschubgeschwindigkeit ausgeführt. (es sei denn, es ist ein adaptiver Vorschub-Override aktiv).

11.6.12. M51 Override-Steuerung der Spindeldrehzahl

- M51 <P1> <$→'- aktiviert die Spindeldrehzahlneufestsetzung für die ausgewählte Spindel. Die Angabe P1 ist optional.
- M51 P0 <$→' - Deaktivieren Sie das Programm zur Neufestsetzung der Spindeldrehzahl.

Wenn diese Funktion deaktiviert ist, hat die Spindeldrehzahlneufestsetzung keinen Einfluss und die Spindeldrehzahl hat den exakten programmierten Wert des S-Wortes (beschrieben im Abschnitt zur Spindle Speed).

The override is also suspended for the duration of a G33 move, and held at the value the cycle started with for the duration of a G76 cycle, whatever M48, M49 or M51 last selected. That selection is restored afterwards.

11.6.13. M52 Adaptive Vorschubsteuerung

- *M52 <P1>* - Verwendung eines adaptiven Vorschubs. Die Angabe P1 ist optional.
- *M52 P0* - Verwendung des adaptiven Feeds beenden.

Wenn der adaptive Vorschub (engl. adaptive feed) aktiviert ist, wird ein externer Eingangswert zusammen mit dem Vorschub-Override-Wert der Benutzeroberfläche und der befohlenen Vorschubgeschwindigkeit verwendet, um die tatsächliche Vorschubgeschwindigkeit einzustellen. Der Wertebereich von motion.adaptive-feed wird durch den Parameter MAX_FEED_OVERRIDE im Abschnitt [DISPLAY] der INI-Datei festgelegt und auf den Bereich −MAX_FEED_OVERRIDE bis +MAX_FEED_OVERRIDE begrenzt. Ein Wert von 0 entspricht einem Feed-Hold.

NOTE Die Verwendung des negativen adaptiven Vorschubs für den Rückwärtslauf ist eine neue Funktion, die noch nicht sehr gut getestet wurde. Sie ist für Plasmaschneider und Drahterodiermaschinen vorgesehen, aber nicht auf diese Anwendungen beschränkt.

11.6.14. M53 Steuerung des Vorschub-Stopps

- *M53 <P1>* - Aktivierung des Vorschubstoppschalters. Der P1 ist optional. Durch die Aktivierung des Vorschubstoppschalters kann die Bewegung mit Hilfe der Vorschubstoppsteuerung unterbrochen werden. In LinuxCNC wird der HAL-Pin *motion.feed-hold* für diesen Zweck verwendet. Ein *true*-Wert wird die Bewegung zu stoppen, wenn *M53* aktiv ist.
- *M53 P0* - deaktiviert den Feed-Stop-Schalter. Der Zustand von *motion.feed-hold* hat keine Auswirkungen auf den Feed, wenn M53 nicht aktiv ist.

11.6.15. M61 Aktuelles Werkzeug einstellen

- *M61 Q-* - ändert die aktuelle Werkzeugnummer, während in MDI-oder Handbetrieb ohne einen Werkzeugwechsel. Wenn Sie LinuxCNC einschalten mit einem Werkzeug bereits in der Spindel, können Sie die Werkzeug-Nummer setzen ohne einen Werkzeugwechsel zu initiieren.

WARNING	Der Werkzeuglängen-Offset wird durch *M61* nicht verändert, verwenden Sie *G43* nach *M61*, um den Werkzeuglängen-Offset zu ändern.

Es ist ein Fehler, wenn:

- Q- ist nicht 0 oder größer

11.6.16. M62 - M65 Digitale Ausgangssteuerung

- *M62 P-* - schaltet den digitalen Ausgang synchron zur Bewegung ein.
- *M63 P-* - Ausschalten des mit der Bewegung synchronisierten digitalen Ausgangs.
- *M64 P-* - Digitalausgang sofort einschalten.
- *M65 P-* - digitale Ausgabe sofort abschalten.

The P- word specifies the digital output number. The P-word ranges from 0 to a default value of 3. If needed the the number of I/O can be increased by using the num_dio parameter when loading the motion controller. See the Motion section for more information. A P-word for an output that the motion controller does not have is an error: the command is refused and a running program is stopped.

Die Befehle M62 und M63 werden in eine Warteschlange gestellt. Nachfolgende Befehle, die sich auf dieselbe Ausgangsnummer beziehen, überschreiben die älteren Einstellungen. Mehr als eine Ausgangsänderung kann durch mehrere M62/M63-Befehle festgelegt werden.

Die tatsächliche Änderung der angegebenen Ausgänge erfolgt zu Beginn des nächsten Fahrbefehls. Wenn es keinen nachfolgenden Fahrbefehl gibt, werden die in der Warteschlange stehenden Ausgangsänderungen nicht ausgeführt. Es ist am besten, immer einen Bewegungs-G-Code (G0, G1, etc.) direkt nach dem M62/63 zu programmieren.

M64 und M65 treten sofort auf, wenn sie von der Bewegungssteuerung empfangen werden. Sie sind nicht mit der Bewegung synchronisiert und unterbrechen den Übergang.

NOTE	M62-65 funktionieren nur dann, wenn die entsprechenden motion.digital-out-*nn*-Pins in Ihrer HAL-Datei mit Ausgängen verbunden sind.

11.6.17. M66 Warten auf Eingabe

```
M66 P- | E- <L->
```

- *P-* - gibt die digitale Eingangsnummer von 0 bis 3 an. (Einstellbar mit motmod Argument num_dio)
- *E-* - gibt die Nummer des Analogeingangs von 0 bis 3 an (einstellbar über das motmod-Argument num_aio)
- *L-* - legt den Wartemodus fest.
  - Mode 0: IMMEDIATE' - kein Warten, kehrt sofort zurück. Der aktuelle Wert des Eingangs wird in Parameter #5399 gespeichert
  - *Mode 1: RISE* - wartet darauf, dass die ausgewählte Eingabe ein RISE-Ereignis ausführt.
  - *Mode 2: FALL* - wartet darauf, dass die ausgewählte Eingabe ein Fallereignis ausführt.
  - *Mode 3: HIGH* - wartet darauf, dass die ausgewählte Eingabe in den HIGH-Zustand wechselt.
  - *Mode 4: LOW* - wartet, bis die ausgewählte Eingabe in den LOW-Zustand wechselt.
- x'Q-' - gibt das Timeout in Sekunden für das Warten an. Wird das Timeout überschritten, so wird die Wartezeit unterbrochen, und die Variable #5399 enthält den Wert -1. Der Q-Wert wird ignoriert, wenn das L-Wort Null ist (SOFORT). Ein Q-Wert von Null ist ein Fehler, wenn das L-Wort ungleich Null ist.
- Für einen Analogeingang ist nur der Modus 0 zulässig.

M66 Beispielzeilen

```
M66 P0 L3 Q5 (bis zu 5 Sekunden warten, bis der digitale Eingang 0 eingeschaltet wird)
```

M66 wartet auf eine Eingabe und stoppt die weitere Ausführung des Programms, bis das gewählte Ereignis (oder der programmierte Timeout) eintritt.

Es ist ein Fehler, M66 sowohl mit einem P-Wort als auch mit einem E-Wort zu programmieren (also sowohl einen analogen als auch einen digitalen Eingang zu wählen). In LinuxCNC sind diese Eingänge nicht in Echtzeit überwacht und sollte daher nicht für zeitkritische Anwendungen verwendet werden.

Die Anzahl der E/A kann durch Verwendung des Parameters num_dio oder num_aio beim Laden des Motion Controllers erhöht werden. Siehe den Abschnitt zu Bewegungen (engl. motion) für weitere Informationen.

NOTE M66 funktioniert nur dann, wenn die entsprechenden motion.digital-in-*nn*-Pins oder motion.analog-in-*nn*-Pins in Ihrer HAL-Datei mit einem Eingang verbunden sind.

Beispiel einer HAL-Verbindung

```
net signal-name motion.digital-in-00 <= parport.0.pin10-in
```

11.6.18. M67 Analogausgang, synchronisiert

```
M67 E- Q-
```

- *M67* - stellt einen mit der Bewegung synchronisierten analogen Ausgang ein.
- *E-* - Ausgangsnummer im Bereich von 0 bis 3 (einstellbar über das motmod-Argument num_aio)
- *Q-* - ist der einzustellende Wert (zum Ausschalten auf 0 setzen).

Die tatsächliche Änderung der angegebenen Ausgänge erfolgt zu Beginn des nächsten Fahrbefehls. Wenn es keinen nachfolgenden Bewegungsbefehl gibt, werden die in der Warteschlange stehenden Ausgangsänderungen nicht durchgeführt. Am besten programmieren Sie immer einen Bewegungs-G-Code (G0, G1, etc.) direkt nach M67. M67 funktioniert genauso wie M62-63.

The number of I/O can be increased by using the num_dio or num_aio parameter when loading the motion controller. See the Motion section for more information. An E-word for an output that the motion controller does not have is an error: the command is refused and a running program is stopped.

NOTE M67 funktioniert nur, wenn die entsprechenden motion.analog-out-*nn* Pins in Ihrer HAL-Datei mit Ausgängen verbunden sind.

11.6.19. M68 Analoger Ausgang, sofort

```
M68 E-Q-
```

- *M68* - sofort einen analogen Ausgang einstellen.
- *E-* - Ausgangsnummer im Bereich von 0 bis 3 (einstellbar über das motmod-Argument num_aio)
- *Q-* - ist der einzustellende Wert (zum Ausschalten auf 0 setzen).

M68-Ausgaben erfolgen sofort, wenn sie vom Motion Controller empfangen werden. Sie sind nicht mit der Bewegung synchronisiert und unterbrechen das Blending. M68 funktioniert genauso wie M64-65.

The number of I/O can be increased by using the num_dio or num_aio parameter when loading the motion controller. See the Motion section for more information. An E-word for an output that the motion controller does not have is an error: the command is refused and a running program is stopped.

NOTE M68 funktioniert nur dann, wenn die entsprechenden motion.analog-out-*nn* Pins in Ihrer HAL-Datei mit Ausgängen verbunden sind.

11.6.20. M70 Modalstatus speichern

Um den modalen Zustand explizit auf der aktuellen Aufrufebene zu speichern, programmieren Sie *M70*. Sobald der modale Zustand mit *M70* gespeichert wurde, kann er durch die Ausführung von *M72* wieder in genau diesen Zustand versetzt werden.

Ein Paar von *M70*- und *M72*-Befehlen wird normalerweise verwendet, um ein Programm gegen

unbeabsichtigte Modaländerungen innerhalb von Unterprogrammen zu schützen.

M70 Gesicherter Zustand

Der gespeicherte Zustand besteht aus:

- aktuelle G20/G21-Einstellungen (imperial/metrisch)
- selected plane (G17/G18/G19 G17.1,G18.1,G19.1)
- Status der Fräserkompensation (G40,G41,G42,G41.1,G42,1)
- Distanzmodus relativ/absolut (G90/G91)
- Vorschubmodus (G93/G94,G95)
- aktuelles Koordinatensystem (G54-G59.3)
- Status der Werkzeuglängenkompensation (G43,G43.1,G49)
- Rückzieh-Modus (G98,G99)
- Spindel-Modus (G96-css oder G97-RPM)
- Bogendistanz-Modus (G90.1, G91.1)
- Radius-/Durchmessermodus der Drehmaschine (G7,G8)
- Pfadsteuerungsmodus (G61, G61.1, G64)
- aktueller Vorschub und Geschwindigkeit (*F*- und *S*-Werte)
- Spindelstatus (M3,M4,M5) - Ein/Aus und Richtung
- Nebel- (M7) und Flut-Kühlung (M8) Status
- Geschwindigkeitsneufestsetzung (M51) und Vorschubneufestsetzung (M50)
- adaptive Vorschubeinstellung (M52)
- Steuerung des Vorschub-Stopps (M53)

Beachten Sie, dass insbesondere der Bewegungsmodus (G1 usw.) NICHT wiederhergestellt wird.

aktuelle Aufruf-Ebene (engl. current call level) bedeutet entweder:

- im Hauptprogramm ausgeführt zu werden. Es gibt einen einzigen Speicherplatz für den Zustand auf der Ebene des Hauptprogramms; wenn mehrere *M70*-Befehle nacheinander ausgeführt werden, wird bei der Ausführung eines *M72* nur der zuletzt gespeicherte Zustand wiederhergestellt.
- innerhalb eines G-Code-Unterprogramms ausgeführt zu werden. Der innerhalb eines Unterprogramms mit *M70* gespeicherte Zustand verhält sich genau wie ein lokaler benannter Parameter - er kann nur innerhalb dieses Unterprogrammaufrufs mit einem *M72* angesprochen werden, und wenn das Unterprogramm beendet wird, verschwindet der Parameter.

Ein rekursiver Aufruf eines Unterprogramms führt eine neue Aufrufebene ein.

11.6.21. M71 Gespeicherten Modalzustand ungültig machen

Der mit einem *M70* oder einem *M73* auf der aktuellen Anrufebene gespeicherte Modalzustand wird

ungültig (kann nicht mehr wiederhergestellt werden).

Ein nachfolgendes *M72* auf der gleichen Aufrufebene schlägt fehl.

Wenn es in einem Unterprogramm ausgeführt wird, das den modalen Zustand durch ein *M73* schützt, wird der modale Zustand durch eine anschließende Rückkehr oder endsub **nicht** wiederhergestellt.

Die Nützlichkeit dieser Funktion ist zweifelhaft. Man sollte sich nicht auf sie verlassen, da sie verschwinden könnte.

11.6.22. M72 Modalen Zustand wiederherstellen

Modaler Zustand, der mit einem *M70* Code gespeichert wurde, kann durch Ausführen eines *M72* wiederhergestellt werden.

Die Handhabung von G20/G21 wird besonders behandelt, da Vorschübe je nach G20/G21 unterschiedlich interpretiert werden: Wenn die Längeneinheiten (mm/in) durch die Wiederherstellungsoperation geändert werden sollen, stellt *M72* zuerst den Abstandsmodus wieder her und dann alle anderen Zustände einschließlich des Vorschubs, um sicherzustellen, dass der Vorschubwert in der richtigen Einheiteneinstellung interpretiert wird.

Es ist ein Fehler, einen *M72* ohne vorherige *M70*-Speicheroperation auf dieser Ebene auszuführen.

Das folgende Beispiel demonstriert das Speichern und explizite Wiederherstellen des modalen Zustands bei einem Unterprogrammaufruf mit *M70* und *M72*. Beachten Sie, dass das Unterprogramm "imperialsub" die M7x-Funktionen nicht "kennt" und unverändert verwendet werden kann:

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsub> sub
g20 (imperial)
g91 (relativer Modus)
F5 (niedriger Vorschub)
S300 (niedrige Drehzahl)
(Debug, im Unterprogramm, Zustand jetzt:)
o<showstate> Aufruf
O<imperialsub> endsub

; Hauptprogramm
G21 (metrisch)
G90 (absolut)
F200 (Schnelle Geschwindigkeit)
S2500 (hohe U/min)

(debug, in main, Zustand jetzt:)
o<showstate> call

M70 (Speichern des Anruferstatus auf globaler Ebene)
O<imperialsub> Aufruf
M72 (Zustand explizit wiederherstellen)
```

```
(Debug, zurück in Main, Zustand jetzt:)
o<showstate> Aufruf
m2
```

11.6.23. M73 Sicherung des Modalzustand und dessen automatische Wiederherstellung (engl. autorestore)

Um den modalen Zustand innerhalb eines Unterprogramms zu speichern und ihn bei *endsub* oder einem beliebigen *Return*-Pfad wiederherzustellen, programmieren Sie *M73*.

Der Abbruch eines laufenden Programms in einem Unterprogramm, das eine *M73*-Operation hat, stellt **nicht** den Zustand wieder her.

Auch das normale Ende (*M2*) eines Hauptprogramms, das ein *M73* enthält, stellt den Zustand **nicht** wieder her.

Die empfohlene Verwendung ist am Anfang eines O-Wort-Unterprogramms wie im folgenden Beispiel. Die Verwendung von *M73* auf diese Weise ermöglicht den Entwurf von Unterprogrammen, die den Modalzustand ändern müssen, schützt aber das aufrufende Programm vor versehentlichen Modaländerungen. Beachten Sie die Verwendung von Vordefinierte benannte Parameter im Unterprogramm "showstate".

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsub> sub
M73 (Zustand des Aufrufers im aktuellen Aufrufkontext speichern, bei Rückkehr oder Endsub
wiederherstellen)
g20 (imperial)
g91 (relativer Modus)
F5 (niedriger Vorschub)
S300 (niedrige Drehzahl)
(debug, im Unterprogramm, Zustand jetzt:)
o<showstate> call

; Hinweis: Hier wird kein M72 benötigt - das folgende endsub oder ein
; explizites 'return' wird den Zustand des Aufrufers wiederherstellen
O<imperialsub> endsub

; Hauptprogramm
g21 (metrisch)
g90 (absolut)
f200 (schnelle Drehzahl)
S2500 (hohe Drehzahl)
(debug, in Hauptprogramm, Zustand jetzt:)
o<showstate> call
o<imperialsub> call
(debug, zurück im Hauptmenü, Status jetzt:)
o<showstate> call
m2
```

11.6.24. M98 und M99

Der Interpreter unterstützt Haupt- und Unterprogramme im Fanuc-Stil mit den M-Codes *M98* und *M99*. Siehe Fanuc-Style Programme.

Selektive Wiederherstellung des modalen Zustands

Die Ausführung eines *M72* oder die Rückkehr aus einer Unterroutine, die ein *M73* enthält, stellt **alle** gesicherten modalen Zustände wieder her.

Wenn nur einige Aspekte des modalen Zustands erhalten bleiben sollen, ist eine Alternative die Verwendung von vordefinierten benannten Parametern, lokalen Parametern und bedingten Anweisungen. Die Idee ist, sich zu Beginn des Unterprogramms die wiederherzustellenden Modi zu merken und diese vor dem Verlassen des Programms wiederherzustellen. Hier ist ein Beispiel, das auf einem Ausschnitt aus *nc_files/tool-length-probe.ngc* basiert:

```
O<Maßnahme> sub (Referenzwerkzeug für Maßnahmen)
;
#<absolute> = #<_absolute> (in lokaler Variable speichern, wenn G90 gesetzt wurde)
;
g30 (über Schalter)
g38.2 z0 f15 (messen)
g91 g0z.2 (außerhalb des Schalters)
#1000=#5063 (Referenzlänge des Werkzeugs speichern)
(print, Referenzlänge ist #1000)
;
O<restore_abs> if [#<absolute>]
    g90 (G90 nur dann wiederherstellen, wenn es bei der Eingabe gesetzt wurde:)
O<restore_abs> endif
;
O<measure> endsub
```

11.6.25. M100-M199 Benutzerdefinierte Befehle

```
M1-- <P- Q->
```

- *M1--* - eine ganze Zahl im Bereich von 100 - 199.
- *P-* - eine Zahl, die als erster Parameter an die Datei übergeben wird.
- *Q-* - eine Zahl, die als zweiter Parameter an die Datei übergeben wird.

NOTE Nach dem Erstellen einer neuen *M1nn*-Datei müssen Sie die grafische Benutzeroberfläche neu starten, damit sie die neue Datei erkennt, sonst erhalten Sie die Fehlermeldung *Unbekannter m-Code*.

Das externe Programm mit den Namen *M100* bis *M199* (keine Erweiterung, ein großes M, das sich im Verzeichnis befindet, auf das der Parameter *[DISPLAY] PROGRAM_PREFIX* der INI-Datei verweist) wird mit den optionalen Werten P und Q als seinen beiden Argumenten ausgeführt.

Die Ausführung der G-Code-Datei wird angehalten, bis das externe Programm beendet wird. Bei anderen Exit-Code des externen Programms als *0*, wird das G-Code Programm gestoppt. Es kann jede gültige ausführbare Datei verwendet werden. Die Datei muss sich in dem Suchpfad befinden, der in der Konfiguration der INI-Datei angegeben ist. Weitere Informationen zu Suchpfaden finden Sie im Abschnitt Display.

Nach dem Erstellen eines neuen M1*nn*-Programms sollte die GUI neu gestartet werden, damit das neue Programm berücksichtigt wird, da sonst ein *Unbekantner M-Code*-Fehler auftritt.

WARNING	Verwenden Sie kein Textverarbeitungsprogramm, um die Dateien zu erstellen oder zu bearbeiten. Ein Textverarbeitungsprogramm hinterlässt unsichtbare Codes, die Probleme verursachen und verhindern können, dass eine Bash- oder Python-Datei funktioniert. Verwenden Sie einen Texteditor wie Geany in Debian oder Notepad++ in anderen Betriebssystemen, um die Dateien zu erstellen oder zu bearbeiten.

Der Fehler "Unbekannter M-Code verwendet" bedeutet eine der folgenden Möglichkeiten:

- Der angegebene benutzerdefinierte Befehl existiert nicht.
- Die Datei ist keine ausführbare Datei.
- Der Dateiname hat eine Erweiterung.
- Der Dateiname entspricht nicht diesem Format Mnnn, wobei nnn = 100 bis 199.
- Der Dateiname enthielt ein kleines M.

Beispiel: Öffnen und Schließen eines Spannzangenverschlusses, der über einen Pin der parallelen Schnittstelle gesteuert wird, mit einer Bash-Skriptdatei unter Verwendung von M101 und M102. Erstellen Sie zwei Dateien mit den Namen M101 und M102. Setzen Sie sie als ausführbare Dateien (typischerweise Rechtsklick/Eigenschaften/Berechtigungen), bevor Sie LinuxCNC ausführen. Stellen Sie sicher, dass der Parallelport-Pin nicht mit irgendetwas in einer HAL-Datei verbunden ist.

M101-Beispieldatei

```
#!/bin/bash
# Datei zum Einschalten von Parport Pin 14, um die Spannzange näher zu öffnen
halcmd setp parport.0.pin-14-out True
exit 0
```

M102 Beispieldatei

```
#!/bin/bash
# Datei zum Ausschalten von Parport Pin 14, um die Spannzange näher zu öffnen
halcmd setp parport.0.pin-14-out False
exit 0
```

Um eine Variable an eine M1nn-Datei zu übergeben, verwenden Sie die Optionen P und Q wie folgt:

```
M100 P123.456 Q321.654
```

M100 Beispieldatei

```
#!/bin/bash
voltage=$1
feedrate=$2
halcmd setp thc.voltage $voltage
halcmd setp thc.feedrate $feedrate
exit 0
```

Um eine grafische Nachricht anzuzeigen und anzuhalten, bis das Meldungsfenster geschlossen wird, verwenden Sie ein grafisches Anzeigeprogramm wie Eye of Gnome, um die Grafikdatei anzuzeigen. Wenn Sie es schließen, wird das Programm fortgesetzt.

M110 Beispieldatei

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png
exit 0
```

Um eine grafische Meldung anzuzeigen und die Bearbeitung der G-Code-Datei fortzusetzen, fügen Sie dem Befehl ein kaufmännisches Und hinzu.

M110 Beispielanzeige und Weiterfahrt

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png &
exit 0
```

11.7.O Codes

11.7.1. Verwendung von O-Codes

O-Codes sorgen für die Ablaufsteuerung in NC-Programmen. Jeder Satz hat eine zugehörige Nummer, die nach dem O verwendet wird. Es muss darauf geachtet werden, dass die O-Nummern richtig zugeordnet werden. O-Codes verwenden den Buchstaben *O* und nicht die Zahl Null als erstes Zeichen in der Nummer wie O100 oder o100. O-Codes werden auch manchmal o-Worte (engl. o-words) genannt und diese Begriffe sind austauschbar.

NOTE Die Verwendung des Kleinbuchstaben *o* erleichtert die Unterscheidung von einer 0, die möglicherweise falsch geschrieben wurde. Zum Beispiel ist bei *o100* leichter zu erkennen als *O100*, dass es sich nicht um eine *0* handelt.

11.7.2. Nummerierung

There are two categories of o-codes with different scoping rules:

Subroutine definitions (`sub`/`endsub`, `call`) are **global**. Each subroutine must have a unique number or name across the entire program and all called files.

Control flow (`if`/`endif`, `while`/`endwhile`, `do`/`while`, `repeat`/`endrepeat`, `break`, `continue`) are **local** to the subroutine or main program where they appear. The interpreter automatically scopes them, so `o100 if` inside `o<helper>` does not conflict with `o100 if` in the main program or in any other subroutine. You can safely reuse the same o-numbers for control flow in different subroutines.

Within a single subroutine body (or the main program), each o-number should still be used for only one control flow block.

Beispiel für eine Nummerierung

```
(the start of o100)
o100 sub
(notice that the if-endif block uses a different number within this sub)
  (the start of o110)
  o110 if [#2 GT 5]
    (some code here)
  (the end of o110)
  o110 endif
  (some more code here)
(the end of o100)
o100 endsub
```

Reusing Control Flow Numbers Across Subroutines (valid)

```
(o100 if in helper.ngc does not conflict with o100 if in another.ngc)

(file: helper.ngc)
o<helper> sub
  o100 if [#1 GT 5]
    (do something)
  o100 endif
o<helper> endsub

(file: another.ngc)
o<another> sub
  o100 if [#1 LT 0]
    (do something else)
  o100 endif
o<another> endsub
```

11.7.3.Kommentare

Kommentare in der gleichen Zeile wie das o-Wort sollten nicht verwendet werden, da sich das Verhalten in Zukunft ändern kann.

Das Verhalten ist undefiniert, wenn:

- The same number is used for more than one block within the same scope (see Nummerierung for scoping rules).
- Andere Wörter in einer Zeile mit einem o-Wort verwendet werden.
- Kommentare in einer Zeile mit einem o-Wort verwendet werden.

11.7.4.Unterroutinen

Unterroutinen beginnen mit *oNNN sub* und enden mit *oNNN endsub*. Die Zeilen zwischen *oNNNsub* und *oNNN endsub* werden nicht ausgeführt, bis das Unterprogramm mit *oNNN call* aufgerufen wird. Jede Subroutine muss eine eindeutige Nummer verwenden.

Beispiel für ein Unterprogramm

```
o100 unter
  G53 G0 X0 Y0 Z0 (Eilgang zum Referenzpunkt der Maschine)
o100 Endsub

(das Unterprogramm wird aufgerufen)
o100 call
M2
```

Weitere Informationen finden Sie in den Abschnitten G53, G0 und M2.

O- Return (engl. für Rücksprung oder Rückkehr)

Innerhalb einer Subroutine kann *o- return* ausgeführt werden. Damit kehrt man sofort zum aufrufenden Code zurück, so als wäre man auf *o- endsub* gestoßen.

O- Return-Beispiel

```
o100 sub
  (Prüfung, ob Parameter #2 größer als 5 ist)
  o110 if [#2 GT 5]
    (Rückkehr zum Anfang des Unterprogramms, wenn der Test wahr ist)
    o100 return
  o110 endif
    (dies wird nur ausgeführt, wenn Parameter #2 nicht größer als 5 ist)
    (DEBUG, parameter 1 ist [#1])
o100 endsub
```

Weitere Informationen finden Sie in den Abschnitten Binäre Operatoren und Parameter.

O- Call (engl. für Aufruf)

o- Call nimmt bis zu 30 optionale Argumente auf, die als *#1*, *#2* , ..., #N an das Unterprogramm übergeben werden. Die Parameter von #N+1 bis #30 haben den gleichen Wert wie im aufrufenden Kontext. Bei der Rückkehr aus dem Unterprogramm werden die Werte der Parameter #1 bis #30 (unabhängig von der Anzahl der Argumente) auf die Werte zurückgesetzt, die sie vor dem Aufruf hatten. Die Parameter #1 - #30 sind lokal für das Unterprogramm.

Da *1 2 3* als die Zahl 123 geparst wird, müssen die Parameter in eckige Klammern gesetzt werden. Im Folgenden wird eine Unterroutine mit 3 Argumenten aufgerufen:

O- Call-Beispiel

```
o100 sub
  (Prüfung, ob Parameter #2 größer als 5 ist)
  o110 if [#2 GT 5]
    (Rückkehr zum Anfang des Unterprogramms, wenn der Test wahr ist)
    o100 return
```

```
  o110 endif
    (dies wird nur ausgeführt, wenn Parameter #2 nicht größer als 5 ist)
    (DEBUG, Parameter 1 ist [#1])
    (DEBUG, Parameter 3 ist [#3])
o100 endif

o100 call [100] [2] [325]
```

Subroutine **definitions** may not be nested. Defining a subroutine inside another subroutine is not allowed and will produce an error. For example, the following is **invalid**:

Invalid Nested Subroutine Definition (produces an error)

```
o<outer> sub
  o100 sub       (INVALID: nested subroutine definition)
    (code here)
  o100 endsub
o<outer> endsub
```

However, **calling** a subroutine from within another subroutine is perfectly valid. It is the `sub`/`endsub` definition that cannot be nested, not the `call`:

Valid Nested Subroutine Call

```
o<outer> sub
  (code here)
  o<helper> call [1] [2]  (OK: calling another sub)
o<outer> endsub
```

Other O-code control flow such as `if`/`endif`, `while`/`endwhile`, `do`/`while`, and `repeat`/`endrepeat` may be used freely inside subroutines.

Subroutines may only be called after they are defined. They may be called from other functions, and may call themselves recursively if it makes sense to do so. The maximum subroutine nesting level is 10.

Unterprogramme können den Wert von Parametern oberhalb von #30 ändern und diese Änderungen werden für den aufrufenden Code sichtbar sein. Unterprogramme können auch den Wert von global named parameters (d.h. Parameter, deren Namen mit dem Unterstrich "_" beginnen) ändern.

Nummerierte Programme im Fanuc-Stil

Nummerierte Programme (sowohl Haupt- als auch Unterprogramme), die M-Codes *M98* call und *M99* return und ihre jeweiligen semantischen Unterschiede sind eine Alternative zu den oben beschriebenen rs274ngc-Unterprogrammen, die aus Gründen der Kompatibilität mit Fanuc- und anderen Maschinensteuerungen bereitgestellt werden.

Nummerierte Programme sind standardmäßig aktiviert und können durch Einfügen von `DISABLE_FANUC_STYLE_SUB = 1` in den Abschnitt `[RS274NGC]` der INI-Datei deaktiviert werden.

NOTE	Nummerierte Haupt- und Unterprogrammdefinitionen und -aufrufe unterscheiden sich sowohl in der Syntax als auch in der Ausführung vom traditionellen rs274ngc. Um

> Verwechslungen vorzubeugen, gibt der Interpreter eine Fehlermeldung aus, wenn Definitionen eines Stils mit Aufrufen eines anderen Stils vermischt werden.

Nummeriertes Unterprogramm Einfaches Beispiel

```
o1 (Beispiel 1) ; Hauptprogramm 1, "Beispiel 1"
M98 P100 ; Unterprogramm 100 aufrufen
M30 ; Hauptprogramm beenden

o100 ; Beginn des Unterprogramms 100
  G53 G0 X0 Y0 Z0 ; Eilgang zum Referenzpunkt
M99 ; Rückkehr aus Unterprogramm 100
```

o1 (Titel)

Der optionale Hauptprogramm-Anfangsblock gibt dem Hauptprogramm die Nummer `1`. Einige Steuerungen behandeln einen optionalen, in Klammern gesetzten Kommentar als Programmtitel, "Beispiel 1" in diesem Beispiel, aber dies hat keine besondere Bedeutung im rs274ngc-Interpreter.

M98 P- <L->

Aufruf eines nummerierten Unterprogramms. Der Satz `M98 P100` ist analog zur traditionellen `o100 call`-Syntax, darf aber nur zum Aufruf eines folgenden nummerierten Unterprogramms verwendet werden, das mit `o100`...`M99` definiert wurde. Ein optionales *L*-Wort legt eine Schleifenanzahl fest.

M30

Das Hauptprogramm muss mit `M02` oder `M30` (oder `M99`; siehe unten) beendet werden.

O- Beginn der Unterprogrammdefinition

Markiert den Beginn einer nummerierten Unterprogrammdefinition. Der Block `o100` ist ähnlich wie `o100 sub`, außer dass er später in der Datei platziert werden muss als der aufrufende Block `M98 P100`.

M99 Rückkehr aus nummeriertem Unterprogramm

Der Block `M99` ist analog zur traditionellen `o100 endsub`-Syntax, darf aber nur ein nummeriertes Programm beenden (`o100` in diesem Beispiel) und darf nicht ein Unterprogramm beenden, das mit der `o100 sub`-Syntax beginnt.

Der Unterprogrammaufruf `M98` unterscheidet sich vom rs274ngc `o call`in folgender Weise:

- Das nummerierte Unterprogramm muss in der Programmdatei auf den `M98`-Aufruf folgen. Der Interpreter gibt einen Fehler aus, wenn das Unterprogramm vor dem Aufrufblock steht.
- Die Parameter `#1`, `#2`, ..., `#30` sind global und in nummerierten Unterprogrammen zugänglich, ähnlich wie höher nummerierte Parameter in Aufrufen im traditionellen Stil. Änderungen an diesen Parametern innerhalb eines Unterprogramms sind globale Änderungen, die nach der Rückkehr des Unterprogramms bestehen bleiben.
- M98"-Unterprogrammaufrufe haben keinen Rückgabewert.
- M98"-Unterprogrammaufrufblöcke können ein optionales L-Wort enthalten, das eine Schleifenwiederholungszahl angibt. Ohne das L-Wort wird das Unterprogramm nur einmal ausgeführt (äquivalent zu `M98 L1`). Ein `M98 L0`-Satz führt das Unterprogramm nicht aus.

In seltenen Fällen kann der M-Code M99 zur Beendigung des Hauptprogramms verwendet werden, wo er ein *Endlosprogramm* anzeigt. Wenn der Interpreter ein M99 im Hauptprogramm erreicht, springt er an den Anfang der Datei zurück und setzt die Ausführung bei der ersten Zeile fort. Ein Beispiel für die Verwendung eines Endlosprogramms ist ein Aufwärmzyklus einer Maschine; ein M30-Satz zum Löschen des Programmendes könnte verwendet werden, um den Zyklus an einem sauberen Punkt zu beenden, wenn der Bediener bereit ist.

Vollständiges Beispiel für ein nummeriertes Unterprogramm

```
o1 ; Hauptprogramm 1
  #1 = 0
  (PRINT,X MAIN BEGIN: 1=#1)
  M98 P100 L5 ; Unterprogramm 100 aufrufen
  (PRINT,X MAIN END: 1=#1)
M30 ; Hauptprogramm beenden

o100 ; Unterprogramm 100
  #1 = [#1 + 1]
  M98 P200 L5 ; Aufruf des Unterprogramms 200
  (PRINT,>> o100: #1)
M99 ; Rückkehr aus Unterprogramm 100

o200 ; Unterprogramm 200
  #1 = [#1 + 0.01]
  (PRINT,>>>>> o200: #1)
M99 ; Rückkehr aus Unterprogramm 200
```

In diesem Beispiel wird der Parameter #1 auf 0 initialisiert. Das Unterprogramm o100 wird fünfmal in einer Schleife aufgerufen. Innerhalb jedes Aufrufs von "o100 wird das Unterprogramm o200 fünfmal in einer Schleife aufgerufen, also insgesamt 25-mal.

Beachten Sie, dass der Parameter #1 global ist. Am Ende des Hauptprogramms, nach den Aktualisierungen innerhalb von o100 und o200, wird sein Wert gleich 5.25 sein.

11.7.5.Wiederholungen (engl. looping)

Die *while-Schleife* hat zwei Strukturen: *while/endwhile*, und *do/while*. In beiden Fällen wird die Schleife verlassen, wenn die *while* Bedingung als falsch bewertet wird. Der Unterschied besteht darin, wann die Testbedingung erfüllt ist. Die *do/while* Schleife führt den Code in der Schleife aus und überprüft dann die Testbedingung. Die *while/endwhile* Schleife führt zuerst den Test durch.

While Endwhile Beispiel

```
(eine Sägezahnform zeichnen)
G0 X1 Y0 (auf Startposition fahren)
#1 = 0 (Parameter #1 den Wert 0 zuweisen)
F25 (Vorschubgeschwindigkeit einstellen)
o101 while [#1 LT 10]
  G1 X0
  G1 Y[#1/10] X1
  #1 = [#1+1] (Inkrementieren des Testzählers)
o101 endwhile
```

```
M2 (Programm beenden)
```

Do While-Beispiel

```
#1 = 0 (Parameter #1 wird der Wert 0 zugewiesen)
o100 do
  (Fehlersuche, Parameter 1 = #1)
  o110 if [#1 EQ 2]
    #1 = 3 (weisen Sie dem Parameter #1 den Wert 3 zu)
    (msg, #1 wurde der Wert 3 zugewiesen)
    o100 continue (zum Anfang der Schleife springen)
  o110 endif
  (hier etwas Code)
  #1 = [#1 + 1] (Inkrementieren des Testzählers)
o100 while [#1 LT 3]
(msg, Schleife fertig!)
M2
```

Innerhalb einer while-Schleife wird mit *o- break* die Schleife sofort verlassen, und mit *o- continue* wird sofort zur nächsten Auswertung der *while* Bedingung übergegangen. Wenn sie immer noch wahr ist, beginnt die Schleife wieder von vorne. Wenn sie falsch ist, wird die Schleife verlassen.

11.7.6.Bedingte Anweisungen und Verzweigungen

Die *if* Bedingung besteht aus einer Gruppe von Anweisungen mit der gleichen *o* Nummer, die mit *if* beginnen und mit *endif* enden. Optionale *elseif* und *else* Bedingungen können zwischen dem beginnenden *if* und dem endenden *endif* stehen.

Wenn die *if* Bedingung als wahr bewertet wird, dann wird die Gruppe von Anweisungen nach der *if* bis zur nächsten Bedingungszeile ausgeführt.

Wenn die *if* Bedingung als falsch bewertet wird, werden die *elseif* Bedingungen der Reihe nach ausgewertet, bis eine als wahr bewertet wird. Wenn die *elseif* Bedingung wahr ist, werden die auf die *elseif* folgenden Anweisungen bis zur nächsten Bedingungszeile ausgeführt. Wenn keine der *if* oder *elseif* Bedingungen als wahr ausgewertet wird, werden die auf die *else* folgenden Anweisungen ausgeführt. Wenn eine Bedingung zu wahr ausgewertet wird, werden keine weiteren Bedingungen in der Gruppe ausgewertet.

If Endif Beispiel

```
(if parameter #31 is equal to 3 set S2000)
o101 if [#31 EQ 3]
  S2000
o101 endif
```

If ElseIf else EndIf-Beispiel

```
(if parameter #2 is greater than 5 set F100)
o102 if [#2 GT 5]
  F100
o102 elseif [#2 LT 2]
(else if parameter #2 is less than 2 set F200)
```

```
  F200
(else if parameter #2 is 2 through 5 set F150)
o102 else
  F150
o102 endif
```

Mehrere Bedingungen können durch *elseif* Anweisungen getestet werden, bis der *else* Pfad schließlich ausgeführt wird, wenn alle vorangegangenen Bedingungen falsch sind:

If elseif else endif-Beispiel

```
(wenn Parameter #2 größer als 5 ist, F100 einstellen)
o102 wenn [#2 GT 5]
  F100
(wenn Parameter #2 kleiner als 2 ist, wird F200 gesetzt)
o102 elseif [#2 LT 2]
  F20
(Parameter #2 liegt zwischen 2 und 5)
o102 else
  F200
o102 endif
```

11.7.7.Wiederholung (engl. repeat)

Mit *repeat* werden die Anweisungen innerhalb von repeat/endrepeat die angegebene Anzahl von Malen ausgeführt. Das Beispiel zeigt, wie Sie eine diagonale Reihe von Formen fräsen könnten, beginnend an der aktuellen Position.

Beispiel mit repeat

```
(5 diagonale Formen fräsen)
G91 (Inkremental-Modus)
o103 repeat [5]
... (Fräscode hier einfügen)
G0 X1 Y1 (diagonale Bewegung zur nächsten Position)
o103 endrepeat
G90 (Absoluter Modus)
```

11.7.8.Indirektion

Die o-Zahl kann durch einen Parameter und/oder eine Berechnung angegeben werden.

Beispiel für Indirektion

```
o[#101+2] call
```

Berechnung von Werten in O-Wörtern

Weitere Informationen zur Berechnung von Werten finden Sie in den folgenden Abschnitten:

- Parameter
- Ausdrücke

- Binäre Operatoren
- Funktionen

11.7.9.Dateien aufrufen (engl. calling files)

To call a separate file with a subroutine name the file the same as your call and include a sub and endsub in the file. The file must be in the directory pointed to by *PROGRAM_PREFIX* or *SUBROUTINE_PATH* in the INI file. The file name can include **lowercase** letters, numbers, dash, and underscore only. A named subroutine file can contain only a single subroutine definition. Do not place additional numbered subroutine definitions (e.g. `o100 sub`) in the same file as a named subroutine. Numbered subroutines share a global namespace and can silently conflict across files. If you need helper subroutines, place each one in its own file.

Beispiel für eine benannte Datei

```
o<myfile> call
```

Beispiel für eine nummerierte Datei

```
o123 call
```

In der aufgerufenen Datei müssen die oxxx sub und endsub enthalten sein und die Datei muss eine gültige Datei sein.

Beispiel für eine aufgerufene Datei

```
(filename myfile.ngc)
o<myfile> sub
  (code here)
o<myfile> endsub
M2
```

NOTE Die Dateinamen sind nur Kleinbuchstaben, so dass *o<MyFile>* vom Interpreter in *o<myfile>* umgewandelt wird. Weitere Informationen über den Suchpfad und Optionen für den Suchpfad finden Sie im Abschnitt zur INI-Konfiguration.

11.7.10.Unterprogramm Rückgabewerte

Unterprogramme können optional einen Wert durch einen optionalen Ausdruck in einer *endsub* oder *return* Anweisung zurückgeben.

Beispiel für einen Rückgabewert

```
o123 return [#2 *5]
...
o123 endsub [3 * 4]
```

Der Rückgabewert eines Unterprogramms wird im *<_value>* *predefined named parameter* *gespeichert, und der<_value_returned>* vordefinierte Parameter wird auf 1 gesetzt, um anzuzeigen, dass ein Wert

zurückgegeben wurde. Beide Parameter sind global und werden kurz vor dem nächsten Unterprogrammaufruf gelöscht.

11.7.11.Fehler

Die folgenden Anweisungen führen zu einer Fehlermeldung und zum Abbruch des Interpreters:

- ein "return" oder "endsub", außerhalb einer Unterdefinition
- ein Label (benamte Markierung) für `repeat`, das an anderer Stelle definiert ist
- ein Label für `while`, das an anderer Stelle definiert ist und sich nicht auf ein `do` bezieht
- ein an anderer Stelle definiertes Label für `if`
- ein undefiniertes Label bei `else` oder `elseif`
- ein Label auf `else`, `elseif` oder `endif`, das nicht auf ein passendes `if` verweist
- eine Bezeichnung für `break` oder `continue`, die nicht auf ein passendes `while` oder `do` verweist
- eine Bezeichnung für `endrepeat` oder ``endwhile`, die nicht auf ein entsprechendes `while` oder `repeat` verweist
- a subroutine definition (`sub`) inside another subroutine body (nested definitions)
- a numbered subroutine called from within a named subroutine file but not found in the offset table or as a separate file

Um diese Fehler zu nicht-fatalen Warnungen auf stderr zu machen, setzen Sie Bit 0x20 bei der Option `[RS274NGC]FEATURE=` in mask ini.

11.8. Andere Codes

11.8.1. F: Vorschubgeschwindigkeit einstellen

Fx - stellt die Vorschubgeschwindigkeit auf *x* ein. *x* ist normalerweise in Maschineneinheiten (Zoll oder Millimeter) pro Minute.

Die Anwendung des Vorschubs ist wie im dedizierten Abschnitt zum Vorschub beschrieben, es sei denn *inverse time feed rate mode* oder *feed per revolution mode* sind in Kraft, in welchem Fall der Vorschub wie im G93 G94 G95 Abschnitt beschrieben ist.

11.8.2. S: Spindeldrehzahl einstellen

Sx [$n] - setzt die Spindeldrehzahl auf *x* Umdrehungen pro Minute (U/min, engl. RPM) mit dem optionalen $ setzt die Spindeldrehzahl für eine bestimmte Spindel. Ohne das $ wird der Befehl standardmäßig auf spindle.0 gesetzt.

Die Spindel(n) oder die ausgewählte Spindel dreht sich mit dieser Geschwindigkeit, wenn ein *M3* oder *M4* in Kraft ist. Es ist OK, ein S-Wort zu programmieren, egal ob sich die Spindel dreht oder nicht. Wenn

der Geschwindigkeits-Override-Schalter aktiviert und nicht auf 100% eingestellt ist, weicht die Geschwindigkeit von der programmierten ab.

Es ist OK, S0 zu programmieren, die Spindel wird sich dann nicht drehen.

Es ist ein Fehler, wenn:

- die S-Nummer negativ ist.

Wie im Abschnitt Rechtsgewindebohrzyklus mit Verweilzeit beschrieben, wird, wenn ein *G84* (Gewindebohren) Bohrzyklus aktiv ist und die Drehzahl- und Vorschubpotentiometer sind aktiviert, dasjenige mit der niedrigsten Einstellung verwendet. Die Drehzahl und der Vorschub bleiben synchronisiert. In diesem Fall kann die Drehzahl von der programmierten abweichen, auch wenn das Drehzahlkorrekturpotentiometer auf 100% eingestellt ist.

11.8.3. T: Werkzeug auswählen

Tx - Vorbereitung für den Wechsel zum Werkzeug *x*.

Das Werkzeug wird erst gewechselt, wenn ein *M6* programmiert wird (siehe Abschnitt M6). Das T-Wort kann in der gleichen Zeile wie der *M6* oder in einer vorhergehenden Zeile erscheinen. Es ist in Ordnung, wenn T-Wörter in zwei oder mehr Zeilen erscheinen, ohne dass das Werkzeug gewechselt wird. Nur das letzte T-Wort wird beim nächsten Werkzeugwechsel wirksam.

NOTE Wenn LinuxCNC für einen nicht-zufälligen Werkzeugwechsler konfiguriert ist (siehe den Eintrag für RANDOM_TOOLCHANGER zum EMCIO Abschnitt), *T0* bekommt eine besondere Behandlung: kein Werkzeug wird ausgewählt. Dies ist nützlich, wenn Sie wollen, dass die Spindel nach einem Werkzeugwechsel leer ist.

NOTE Wenn LinuxCNC für einen zufälligen Werkzeugwechsler konfiguriert ist (siehe den Eintrag für RANDOM_TOOLCHANGER zum EMCIO Abschnitt), *T0* bekommt keine besondere Behandlung: T0 ist ein gültiges Werkzeug wie jedes andere. Es ist üblich, T0 auf einem Zufallswerkzeugwechsler zu verwenden, um eine leere Tasche zu verfolgen, so dass er sich wie ein Nicht-Zufallswerkzeugwechsler verhält und die Spindel entlädt.

Es ist ein Fehler, wenn:

- eine negative T-Nummer verwendet wird,
- T Nummer verwendet wird, die nicht in der Werkzeugtabellendatei enthalten ist (mit der Ausnahme, dass T0 bei nicht zufälligen Werkzeugwechslern akzeptiert wird, wie oben erwähnt).

Bei einigen Maschinen bewegt sich das Karussell, wenn ein T-Wort programmiert wird, während gleichzeitig eine Bearbeitung stattfindet. Bei solchen Maschinen spart man Zeit, wenn man das T-Wort mehrere Zeilen vor einem Werkzeugwechsel programmiert. Eine gängige Programmierpraxis für solche Maschinen besteht darin, das T-Wort für das nächste zu verwendende Werkzeug in die Zeile nach einem Werkzeugwechsel zu setzen. Dadurch wird die verfügbare Zeit für die Bewegung des Karussells maximiert.

Eilgangbewegungen nach einem *T<n>* werden in der AXIS-Vorschau erst nach einer Vorschubbewegung angezeigt. Dies gilt für Maschinen, die zum Werkzeugwechsel lange Strecken zurücklegen, wie z. B. eine Drehmaschine. Dies kann anfangs sehr verwirrend sein. Um diese Funktion für das aktuelle Werkzeug auszuschalten, programmieren Sie einen G1 ohne eine Bewegung nach dem *T<n>*.

11.9. G-Code Beispiele

Nach der Installation von LinuxCNC mehrere Beispiel-Dateien sind in der /nc_files Ordner platziert. Stellen Sie sicher, dass die Beispieldatei für Ihre Maschine geeignet ist, bevor Sie sie ausführen.

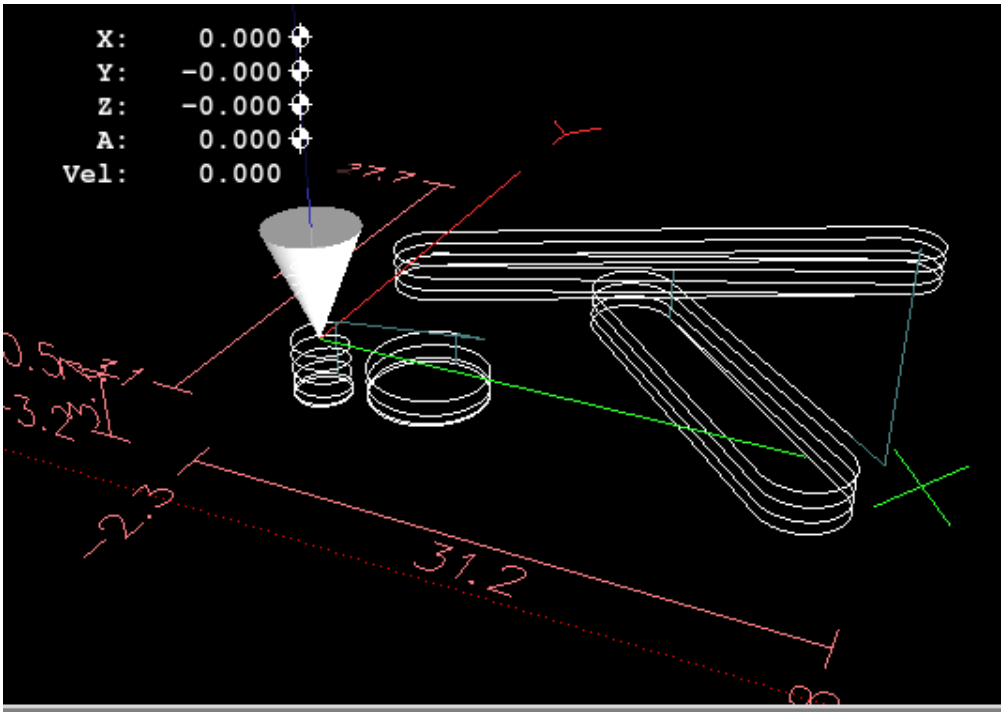
11.9.1. Beispiele für eine Fräsmaschine

Fräsen von Spiralbohrungen

- Dateiname: useful-subroutines.ngc
- Beschreibung: Unterprogramm zum Fräsen einer Bohrung mit Hilfe von Parametern.

Schlitzen (engl. slotting)

- Dateiname: useful-subroutines.ngc
- Beschreibung: Unterprogramm zum Fräsen einer Nut mit Parametern.

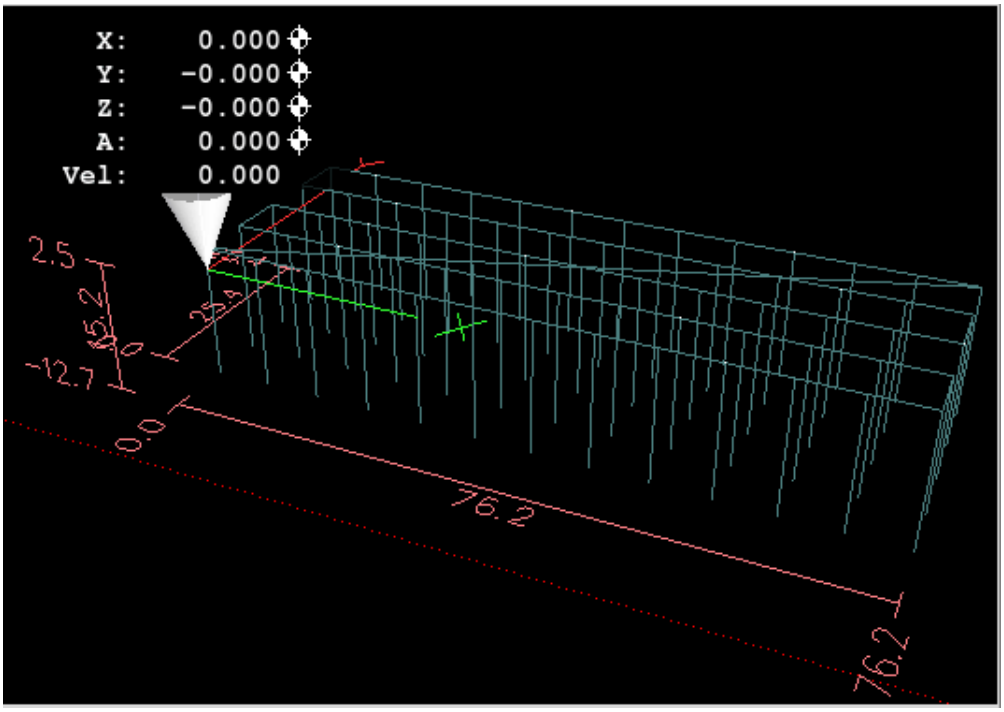


Rastersonde (engl. grid probe)

- Dateiname: gridprobe.ngc
- Beschreibung: Rechteckige Sondierung

Dieses Programm testet wiederholt in einem regelmäßigen XY-Raster und schreibt die getestete Position

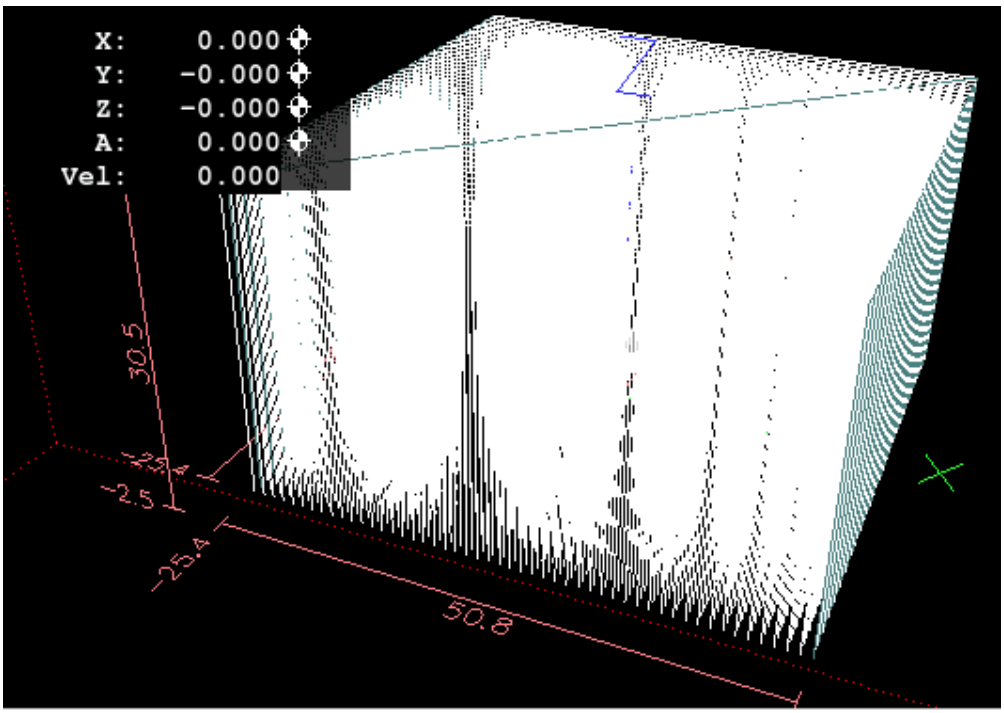
in die Datei *probe-results.txt* im selben Verzeichnis wie die .ini-Datei.



Intelligente Sonde (engl. smart probe)

- Dateiname: smartprobe.ngc
- Beschreibung: Rechteckige Sondierung

Dieses Programm sondiert wiederholt ein regelmäßiges XY-Gitter und schreibt den gesondeten Ort in die Datei *probe-results.txt* im selben Verzeichnis wie die .ini-Datei. Dies ist eine Verbesserung gegenüber der Rastersondierungsdatei.



Werkzeuglängen-Messtaster

- Dateiname: tool-length-probe.ngc
- Beschreibung: Bestimmung der Werkzeuglängen

Dieses Programm zeigt ein Beispiel für die automatische Messung von Werkzeuglängen mit Hilfe eines Schalters, der an den Tastereingang angeschlossen ist. Dies ist nützlich für Maschinen ohne Werkzeughalter, bei denen die Länge eines Werkzeugs jedes Mal anders ist, wenn es eingesetzt wird.

Lochsonde (engl. hole probe)

- Dateiname: probe-hole.ngc
- Beschreibung: Finden des Zentrums und des Durchmessers eines Lochs.

Das Programm demonstriert, wie man den Mittelpunkt eines Lochs findet, den Lochdurchmesser misst und die Ergebnisse aufzeichnet.

Fräserkompensation

- Dateiname: comp-g1.ngc
- Beschreibung: Ein- und Ausfahrbewegungen mit Kompensation des Werkzeugradius.

Dieses Programm demonstriert die Besonderheit des Werkzeugwegs ohne und mit Werkzeugradiuskorrektur. Der Werkzeugradius wird aus der Werkzeugtabelle übernommen.

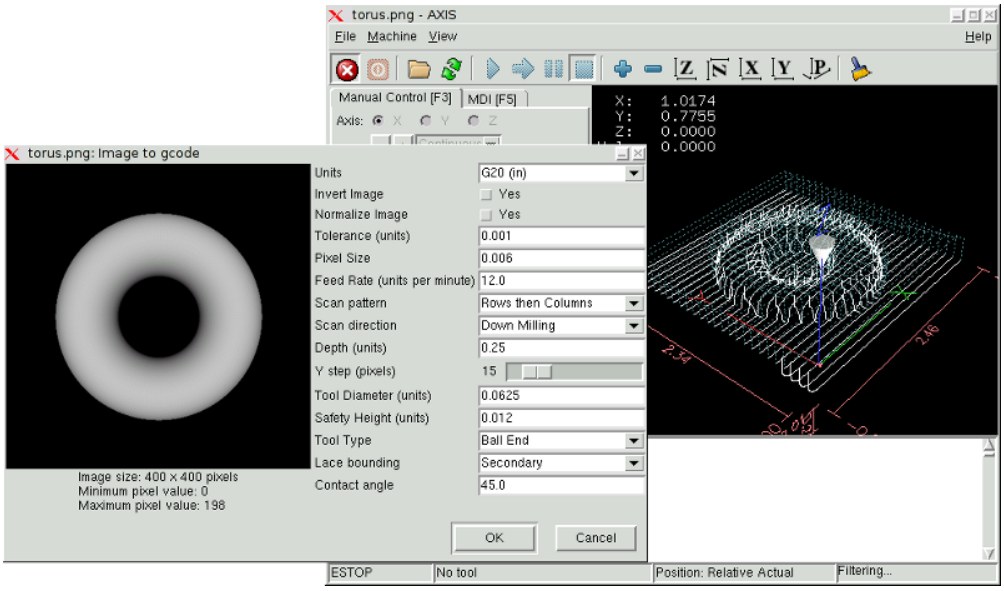
11.9.2. Beispiele für Drehmaschinen

Gewinde-Drehen (engl. threading)

- Dateiname lathe-g76.ngc
- Beschreibung: Plandrehen, Gewindeschneiden und Abstechen.

Diese Datei zeigt ein Beispiel für das Gewindeschneiden auf einer Drehmaschine unter Verwendung von Parametern.

11.10. Vom Bild zu G-Code



11.10.1. Was ist eine Tiefenkarte (engl. depth map)?

Eine Tiefenkarte ist ein Graustufenbild, bei dem die Helligkeit der einzelnen Pixel der Tiefe (oder Höhe) des Objekts an jedem Punkt entspricht.

11.10.2. Integration von Image-to-Gcode in die AXIS-Benutzeroberfläche

Fügen Sie die folgenden Zeilen in den Abschnitt *[FILTER]* Ihrer INI-Datei ein, damit AXIS automatisch image-to-gcode aufruft, wenn Sie ein PNG-, GIF- oder JPG-Bild öffnen:

```
PROGRAM_EXTENSION = .png,.gif,.jpg Grayscale Depth Image
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
```

Die Standard-Konfigurationsdatei *sim/axis.ini* ist bereits auf diese Weise vorbereitet.

11.10.3. Verwendung von `image-to-gcode`

Starten Sie `image-to-gcode` entweder durch Öffnen einer Bilddatei in AXIS oder durch Aufrufen von `image-to-gcode` über das Terminal wie folgt:

```
image-to-gcode torus.png > torus.ngc
```

Überprüfen Sie alle Einstellungen in der rechten Spalte und drücken Sie dann auf OK, um den G-Code zu erstellen. Je nach Bildgröße und gewählten Optionen kann dies einige Sekunden bis einige Minuten dauern. Wenn Sie das Bild in AXIS laden, wird der G-Code automatisch geladen und in der Vorschau angezeigt, sobald die Umwandlung durch `image-to-gcode` (ein Programm, wörtlich engl. "Bild zu G-Code") abgeschlossen ist. Wenn Sie in AXIS auf "Neu laden" klicken, wird der Bildschirm mit den image-to-gcode-Optionen erneut angezeigt, so dass Sie die Optionen anpassen können.

11.10.4. Optionen

Einheiten

Gibt an, ob G20 (Zoll) oder G21 (mm) im generierten G-Code und als Einheiten für jede mit "(units)" bezeichnete Option verwendet werden soll.

Bild invertieren (engl. invert image)

Wenn "nein", ist das schwarze Pixel der niedrigste Punkt und das weiße Pixel der höchste Punkt. Wenn "Ja", ist das schwarze Pixel der höchste Punkt und das weiße Pixel der niedrigste Punkt.

Bild normalisieren (engl. normalize image)

Wenn "ja", wird das dunkelste Pixel auf Schwarz und das hellste Pixel auf Weiß umgestellt.

Erweitern des Bildrandes (engl. expand image border)

Bei *None* (engl. für keinen) wird das Eingabebild so verwendet, wie es ist, und Details, die sich an den Rändern des Bildes befinden, können abgeschnitten werden. Bei *White* (engl. für weiß) oder *black* (engl. für Schwarz) wird an allen Seiten ein Rand aus Pixeln in Höhe des Werkzeugdurchmessers hinzugefügt, und Details an den Rändern des Bildes werden nicht abgeschnitten.

Toleranz (Einheiten)

Wenn eine Reihe von Punkten innerhalb der "Toleranz" (engl. tolerance) auf eine geraden Linie liegen, werden sie als eine gerade Linie ausgegeben. Die Erhöhung der Toleranz kann zu einer besseren Leistung bei der Konturierung in LinuxCNC führen, kann aber auch kleine Details im Bild entfernen oder verwischen.

Pixelgröße (Einheiten)

Ein Pixel im Eingabebild entspricht dieser Anzahl von Einheiten - in der Regel ist diese Zahl viel kleiner als 1,0. Um beispielsweise ein 2,5 x 2,5-Zoll-Objekt aus einer 400 x 400-Bilddatei zu fräsen, verwenden Sie eine Pixelgröße von .00625, da 2,5 / 400 = .00625.

Tauchvorschubgeschwindigkeit (Einheiten pro Minute)

Die Vorschubgeschwindigkeit für die erste Eintauchbewegung.

Vorschubgeschwindigkeit (Einheiten pro Minute)

Die Vorschubgeschwindigkeit für andere Teile der Bahn.

Spindeldrehzahl (RPM)

Der Spindeldrehzahl-S-Code, der in die G-Code-Datei eingefügt werden soll.

Abtastmuster (engl. scan pattern)

Mögliche Scanmuster sind:

- Reihen
- Spalten
- Zeilen, dann Spalten
- Spalten, dann Zeilen

Scanrichtung

Mögliche Scanrichtungen sind:

- Positive (engl. positiv): Starten Sie das Fräsen bei einem niedrigen X- oder Y-Achsenwert und bewegen Sie sich zu einem hohen X- oder Y-Achsenwert.
- Negative (engl. für negativ): Starten Sie das Fräsen bei einem hohen X- oder Y-Achsenwert und bewegen Sie sich zu einem niedrigen X- oder Y-Achsenwert.
- Alternating (engl. für abwechselnd): Beginnen Sie am gleichen Ende der X- oder Y-Achse, an dem die letzte Bewegung endete. Dies reduziert die Anzahl der Verfahrbewegungen.
- Up Milling (engl. für aufwärts fräsen): Beginnen Sie mit dem Fräsen an niedrigen Punkten und bewegen Sie sich zu hohen Punkten.
- Down Milling (engl. für abwärts fräsen): Beginnen Sie das Fräsen an hohen Punkten und bewegen Sie sich zu niedrigen Punkten.

Tiefe (engl. depth) (Einheiten)

Die Materialoberseite ist immer bei *Z = 0*. Der tiefste Schnitt in das Material ist bei *Z = -tiefe* (engl. depth).

Schrittweite (engl. step over) (Pixel)

Der Abstand zwischen benachbarten Zeilen oder Spalten. Um die Anzahl der Pixel für einen gegebenen Einheitsabstand zu ermitteln, berechnen Sie *Abstand/Pixelgröße* und runden Sie auf die nächste ganze Zahl. Wenn zum Beispiel *Pixelgröße=.006* und der gewünschte Step Over *Abstand=.015*, dann verwenden Sie einen Step Over von 2 oder 3 Pixeln, denn *.015/.006=2.5*.

Werkzeug-Durchmesser

Der Durchmesser des schneidenden Teils des Werkzeugs.

Sicherheitshöhe

Die Höhe, die bei Verfahrbewegungen angefahren werden soll. image-to-gcode geht immer davon aus, dass die Oberseite des Materials bei *Z=0* liegt.

Werkzeug-Typ

Die Form des schneidenden Teils des Werkzeugs. Mögliche Werkzeugformen sind:

- Kugelkopf (engl. ball end)
- Flaches Ende (engl. flat end)
- 45 Grad "V" (engl. vee)
- 60 Grad "Vee"

Strukturbegrenzung (engl. lace bounding)

Damit wird gesteuert, ob Bereiche, die entlang einer Zeile oder Spalte relativ flach sind, übersprungen werden. Diese Option ist nur sinnvoll, wenn sowohl Zeilen als auch Spalten gefräst werden. Mögliche Begrenzungsoptionen sind:

- None (engl. für keine): Zeilen und Spalten werden beide vollständig gefräst.
- Secondary (engl. für Sekundär): Beim Fräsen in der zweiten Richtung werden Bereiche, die nicht stark in diese Richtung geneigt sind, übersprungen.
- Voll: Beim Fräsen in der ersten Richtung werden Bereiche, die stark in die zweite Richtung geneigt sind, übersprungen. Beim Fräsen in der zweiten Richtung werden Bereiche, die nicht stark in diese Richtung geneigt sind, übersprungen.

Kontaktwinkel

Wenn die Rasterbegrenzung (oder lieber Spitzenbegrenzung?, engl. lace bounding) nicht "none" (engl. für keine) ist, werden Neigungen, die größer als der "Kontaktwinkel" sind, als "starke" Neigungen und Neigungen, die kleiner als dieser Winkel sind, als schwache Neigungen betrachtet.

Schruppversatz und Schrupptiefe pro Durchgang

Image-to-gcode kann optional Schruppdurchgänge durchführen. Die Tiefe der aufeinanderfolgenden Schruppdurchgänge wird durch "Schrupptiefe pro Durchgang" angegeben. Wenn Sie z. B. 0,2 eingeben, wird der erste Schruppdurchgang mit einer Tiefe von 0,2 durchgeführt, der zweite Schruppdurchgang mit einer Tiefe von 0,4 usw., bis die volle Tiefe des Bildes erreicht ist. Kein Teil eines Schruppdurchgangs schneidet näher als der Schruppversatz zum endgültigen Teil. Die folgende Abbildung zeigt ein hohes vertikales Feature, das gefräst wird. In diesem Bild beträgt die Schrupptiefe pro Durchgang 0,2 Zoll und der Schruppversatz 0,1 Zoll.

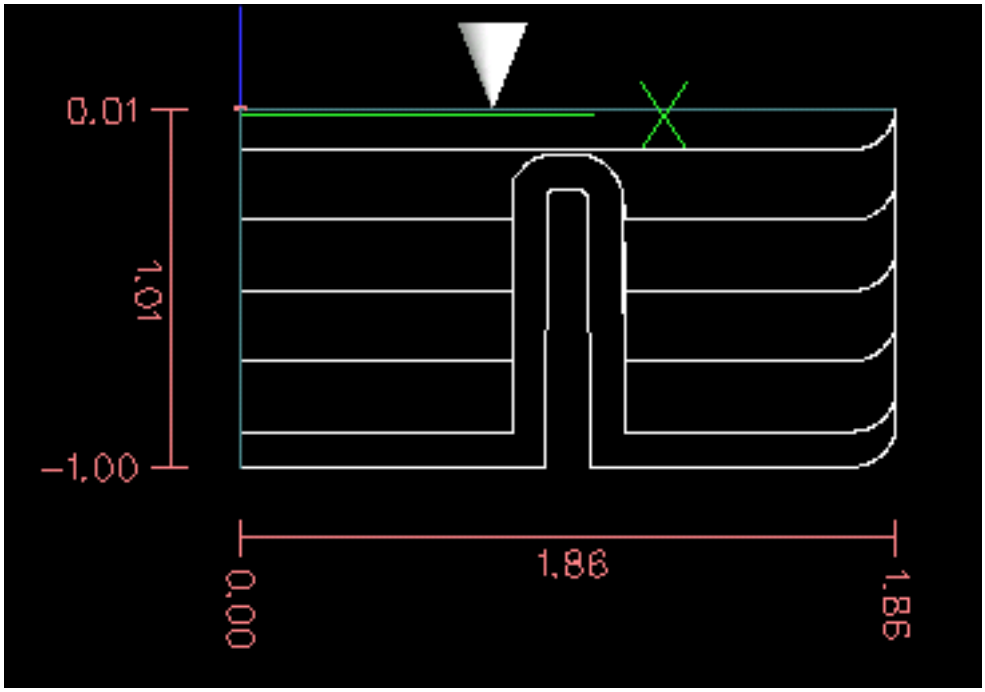


Figure 220. Schruppen und Schlichten

11.11. RS274/NGC Unterschiede

11.11.1. Änderungen gegenüber RS274/NGC

Unterschiede mit Einfluss auf die Interpretation von RS274/NGC-Programmen

Position nach einem Werkzeugwechsel

In LinuxCNC kehrt die Maschine nicht auf seine ursprüngliche Position nach einem Werkzeugwechsel zurück. Diese Änderung wurde vorgenommen, weil das neue Werkzeug länger sein könnte als das alte Werkzeug, und die Bewegung auf die ursprüngliche Maschinenposition könnte daher für die Werkzeugspitze zu niedrig sein.

Offset-Parameter sind Einheiten der INI-Datei

In LinuxCNC werden die Werte in den Parametern für die G28 und G30 Referenzpunkte, die P1...P9 Koordinatensysteme und die G92 Offset sind in "INI-Datei Einheiten" gespeichert. Diese Änderung wurde vorgenommen, weil sonst die Bedeutung eines Standortes geändert, je nachdem, ob G20 oder G21 aktiv war, wenn G28, G30, G10 L2, oder G92.3 programmiert wird.

Längen/Durchmesser der Werkzeugtabelle sind in den in der INI-Datei spezifizierten Einheiten angegeben

In LinuxCNC werden die Werkzeuglängen (Offsets) und Durchmesser in der Werkzeugtabelle nur in der in der INI-Datei spezifizierten Einheiten angegeben. Diese Änderung wurde vorgenommen, da sich sonst die Länge eines Werkzeugs und sein Durchmesser ändern würden, je nachdem, ob G20 oder G21 beim Initiieren der G43-, G41- und G42-Modi aktiv war. Dies machte es unmöglich, G-Code in den nicht-nativen Einheiten der Maschine auszuführen, selbst wenn der G-Code einfach und wohlgeformt war (beginnend mit G20 oder G21 und keine Einheiten im gesamten Programm geändert), ohne die Werkzeugtabelle zu ändern.

G84, G87 nicht implementiert

G84 und G87 sind derzeit nicht implementiert, können aber in einer zukünftigen Version von LinuxCNC hinzugefügt werden.

G28, G30 mit Achswörtern

Wenn G28 oder G30 mit nur einigen vorhandenen Achsenwörtern programmiert wird, bewegt LinuxCNC nur die benannten Achsen. Dies ist bei anderen Maschinensteuerungen üblich. Um einige Achsen zu einem Zwischenpunkt und dann alle Achsen zu dem vordefinierten Punkt zu bewegen, schreiben Sie zwei Zeilen G-Code:

```
G0 X- Y- (Achsen auf Zwischenpunkt fahren)
G28 (alle Achsen auf vordefinierten Punkt fahren)
```

11.11.2. Ergänzungen zu RS274/NGC

Unterschiede ohne Einfluss auf die Bedeutung der RS274/NGC Programme

G33, G76 Gewindecodes

Diese Codes sind in RS274/NGC nicht definiert.

G38.2

Die Tastspitze ist nicht nach einer G38.2 Bewegung zurückgezogen. Dieser Rückzug Bewegung kann in einer zukünftigen Version von LinuxCNC hinzugefügt werden.

G38.3...G38.5

Diese Codes sind in RS274/NGC nicht definiert

O-Codes

Diese Codes sind in RS274/NGC nicht definiert

M50...M53 Neufestsetzungen (engl. overrides)

Diese Codes sind in RS274/NGC nicht definiert

M61..M66

Diese Codes sind in RS274/NGC nicht definiert

G43, G43.1

Negative Werkzeuglängen

In der RS274/NGC-Spezifikation heißt es, dass alle Werkzeuglängen positiv sein sollen. G43 funktioniert jedoch auch bei negativen Werkzeuglängen.

Drehwerkzeuge

Die G43-Werkzeuglängenkompensation kann das Werkzeug sowohl in der X- als auch in der Z-Dimension versetzen. Diese Funktion ist vor allem bei Drehbänken nützlich.

Dynamische Werkzeuglängen

LinuxCNC ermöglicht die Angabe einer berechneten Werkzeuglänge durch G43.1 I K.

G41.1, G42.1

LinuxCNC ermöglicht die Angabe eines Werkzeugdurchmessers und, wenn im Drehmaschinenmodus, Orientierung durch G-Code. Das Format ist G41.1/G42.1 D L, wo D ist der Durchmesser und L (wenn angegeben) ist die Drehmaschine Werkzeug Orientierung.

G43 ohne H-Wort

Mit NGC ist dies nicht erlaubt. In LinuxCNC, setzt es Länge Offsets für die derzeit geladenen Werkzeug. Ist aktuell kein Werkzeug geladen, so ist es ein Fehler. Diese Änderung wurde vorgenommen, damit der Benutzer die Werkzeugnummer nicht an zwei Stellen für jeden Werkzeugwechsel angeben muss, und weil es im Einklang mit der Art und Weise ist wie G41/G42 arbeitet, wenn das D-Wort nicht angegeben ist.

U-, V- und W-Achsen

LinuxCNC ermöglicht Maschinen mit bis zu 9 Achsen durch die Definition einer zusätzlichen Reihe von 3 linearen Achsen bekannt als U, V und W

[1] persistent_range,Der Bereich der persistenten Parameter kann sich mit fortschreitender Entwicklung ändern. Dieser Bereich liegt derzeit zwischen 5161 und 5390. Die im Array *required_parameters* in der Datei src/emc/rs274ngc/interp_array.cc definiert.

[2] Der RS274/NGC-Interpreter verwaltet ein Array mit nummerierten Parametern. Seine Größe wird durch das Symbol *RS274NGC_MAX_PARAMETERS* in der Datei src/emc/rs274ngc/interp_internal.hh) definiert. Diese Anzahl numerischer Parameter kann sich auch erhöhen, wenn die Entwicklung die Unterstützung für neue Parameter hinzufügt.

Chapter 12. Virtuelle Schalttafeln

12.1. PyVCP

12.1.1. Einführung

PyVCP, **Py**thon **V**irtual **C**ontrol **P**anel, wurde entwickelt, um dem Integrator die Möglichkeit zu geben, die AXIS-Schnittstelle mit Schaltflächen und Anzeigen für spezielle Aufgaben anzupassen.

Hardware-Maschinenbedienfelder können eine Menge E/A-Pins belegen und teuer sein. Hier haben virtuelle Control Panels den Vorteil, dass es nichts kostet, ein PyVCP zu erstellen.

Virtuelle Bedienfelder können zum Testen oder Überwachen verwendet werden, um reale E/A-Geräte beim Debuggen der Kontaktplanlogik vorübergehend zu ersetzen oder um ein physisches Bedienfeld zu simulieren, bevor Sie es bauen und mit einer E/A-Platine verbinden.

Die folgende Grafik zeigt viele der PyVCP-Widgets.

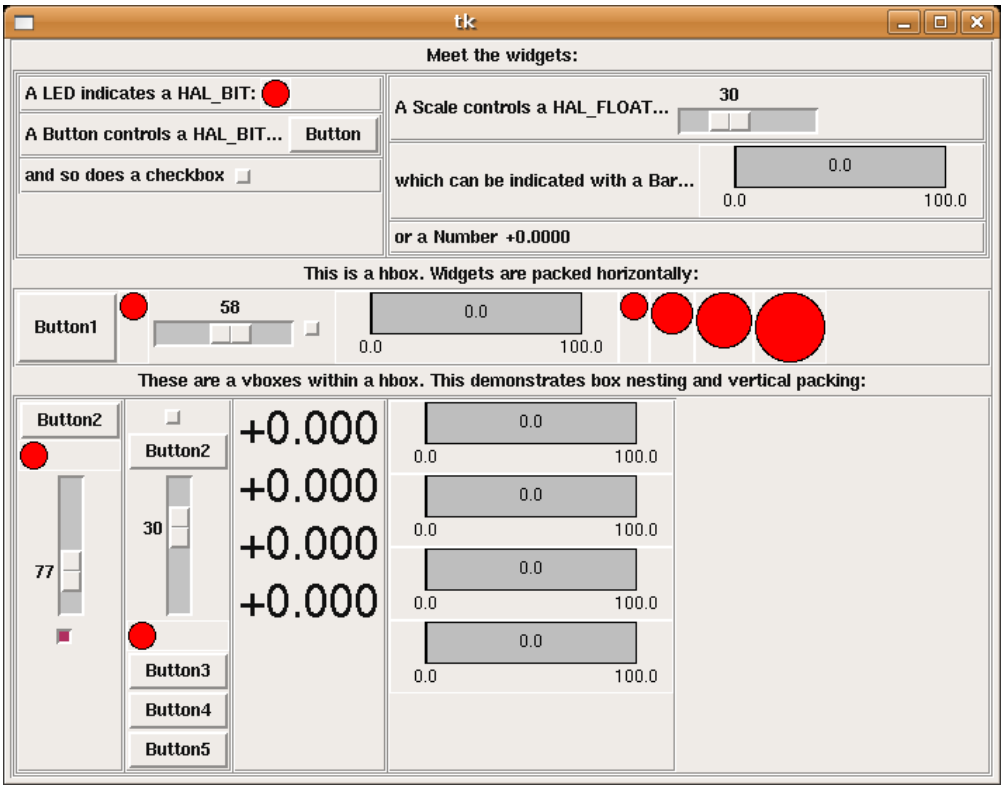


Figure 221. PyVCP Widgets Showcase

12.1.2. Panel Konstruktion

Das Layout eines PyVCP-Panels wird mit einer XML-Datei festgelegt, die Widget-Tags zwischen <pyvcp> und </pyvcp> enthält. Zum Beispiel:

```
<pyvcp>
    <label text="This is a LED indicator"/>
    <led/>
```

```
</pyvcp>
```



Figure 222. Einfaches PyVCP LED-Panel Beispiel

Wenn Sie diesen Text in eine Datei mit dem Namen tiny.xml einfügen und Folgendes ausführen

```
halcmd loadusr pyvcp -c mypanel tiny.xml
```

PyVCP will create the panel for you, which includes two widgets, a Label with the text *This is a LED indicator*, and a LED, used for displaying the state of a HAL BOOL signal. It will also create a HAL component named *mypanel* (all widgets in this panel are connected to pins that start with *mypanel.*). Since no <halpin> tag was present inside the <led> tag, PyVCP will automatically name the HAL pin for the LED widget mypanel.led.0

Für eine Liste der Widgets und ihrer Tags und Optionen, siehe nachfolgende Widget Übersicht.

Sobald Sie Ihr Panel erstellt haben, können Sie mit dem halcmd HAL-Signale mit den PyVCP-Pins verbinden:

```
net <signal-name> <pin-name> <opt-direction> <opt-pin-name>signal-name
```

Wenn Sie mit HAL noch nicht vertraut sind, ist das Kapitel HAL-Grundlagen im Integrator-Handbuch ein guter Ausgangspunkt.

12.1.3. Sicherheit

Teile von PyVCP-Dateien werden als Python-Code ausgewertet und können alle für Python-Programme verfügbaren Aktionen ausführen. Verwenden Sie nur PyVCP-XML-Dateien aus einer Quelle, der Sie vertrauen.

12.1.4. ACHSE

Da AXIS dasselbe GUI-Toolkit (Tkinter) wie PyVCP verwendet, ist es möglich, ein PyVCP-Panel entweder an der rechten Seite oder am unteren Rand der AXIS-Benutzeroberfläche einzubinden. Es ist nicht möglich, ein Panel an beiden Positionen gleichzeitig anzuzeigen. Ein typisches Beispiel wird im Folgenden erläutert.

Zusätzlich oder anstelle der Anzeige eines PyVCP-Panels wie oben beschrieben ist es möglich, ein oder mehrere PyVCP-Panels als eingebettete Registerkarten in der AXIS-GUI anzuzeigen. Dies wird durch den folgenden Eintrag im Abschnitt '[DISPLAY]' der INI-Datei erreicht:

```
EMBED_TAB_NAME    = Spindle
EMBED_TAB_COMMAND = pyvcp spindle.xml
```

Die Textbeschriftung der Registerkarte AXIS zeigt "Spindel" an.

Beispiel-Panel

Legen Sie Ihre PyVCP-XML-Datei, die das Panel beschreibt, in demselben Verzeichnis ab, in dem sich Ihre INI-Datei befindet. Angenommen, wir möchten die aktuelle Spindeldrehzahl mit einem Balken-Widget anzeigen. Fügen Sie das Folgende in eine Datei namens spindle.xml ein:

```
<pyvcp>
    <label>
       <text>"Spindeldrehzahl:"</text>
    </label>
   <bar>
        <halpin>"spindle-speed"</halpin>
        <max_>5000</max_>
    </bar>
</pyvcp>
```

Hier haben wir ein Panel mit einem Label- und einem Balken-Widget erstellt, festgelegt, dass der HAL-Pin, der mit dem Balken verbunden ist, *spindle-speed* heißen soll, und den maximalen Wert des Balkens auf 5000 gesetzt (siehe Widget Übersicht unten für alle Optionen). Um AXIS auf diese Datei aufmerksam zu machen und sie beim Start aufzurufen, müssen wir in der `[DISPLAY]` Sektion der INI-Datei folgendes angeben:

```
PYVCP = spindle.xml
```

Wenn das Panel am unteren Rand der AXIS-Benutzeroberfläche erscheinen soll, müssen wir im Abschnitt `[DISPLAY]` der INI-Datei Folgendes angeben:

```
PYVCP_POSITION = BOTTOM
```

Alles andere als BOTTOM oder das Weglassen dieser Variable platziert das PyVCP-Panel auf der rechten Seite.

Damit unser Widget tatsächlich die Spindel-Geschwindigkeit anzeigt, muss es mit dem entsprechenden HAL-Signal verbunden werden. Eine HAL-Datei, die ausgeführt wird, sobald AXIS und PyVCP gestartet sind, kann im Abschnitt [HAL] der INI-Datei angegeben werden:

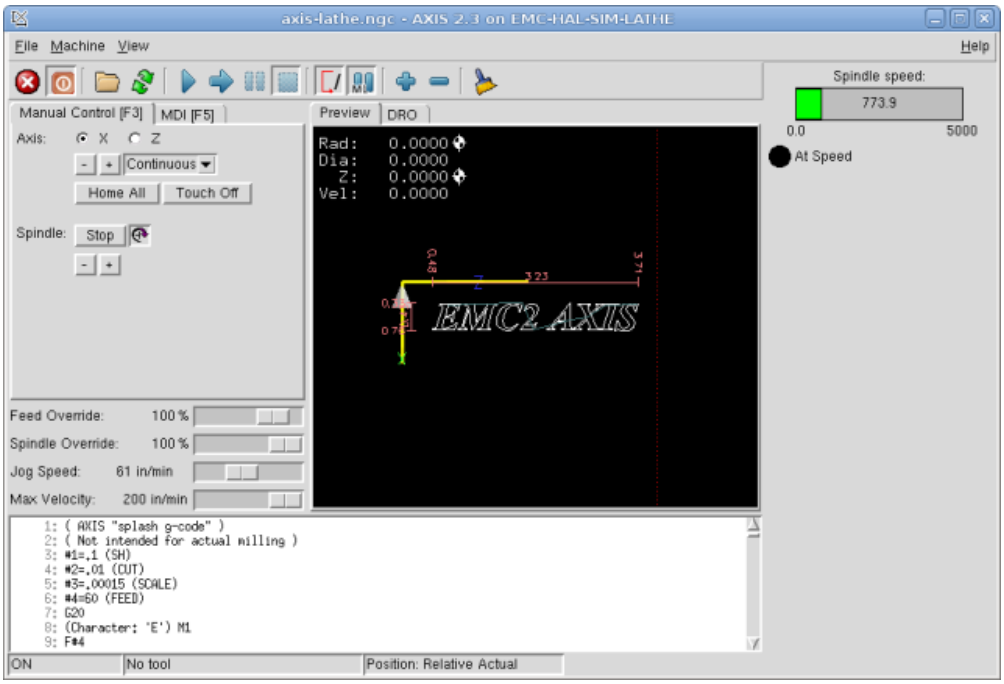
```
POSTGUI_HALFILE = spindle_to_pyvcp.hal
```

Diese Änderung führt die in *spindle_to_pyvcp.hal* angegebenen HAL-Befehle aus. In unserem Beispiel könnte der Inhalt wie folgt aussehen:

```
net spindle-rpm-filtered => pyvcp.spindle-speed
```

unter der Annahme, dass ein Signal namens *spindle-rpm-filtered* bereits existiert. Beachten Sie, dass in Verbindung mit AXIS alle PyVCP-Panel-Widget-HAL-Pins Namen haben, die mit *pyvcp.* beginnen, alle PyVCP-Embedded-Tab-Widget-HAL-Pins beginnen mit dem als EMBED_TAB_NAME angegebenen Namen,

der in Kleinbuchstaben umgewandelt wurde.



So sollte das neu erstellte PyVCP-Panel in AXIS aussehen. Die Konfiguration "sim/lathe" ist bereits auf diese Weise konfiguriert.

12.1.5. Eigenständig (engl. stand alone)

Dieser Abschnitt beschreibt, wie PyVCP-Panels mit oder ohne LinuxCNCs Maschinensteuerung angezeigt werden können.

Um ein eigenständiges PyVCP-Panel mit LinuxCNC zu laden, verwenden Sie diese Befehle:

```
loadusr -Wn mypanel pyvcp -g WxH+X+Y -c mypanel <path/>panel_file.xml
```

Sie würden dies verwenden, wenn Sie ein schwebendes Bedienfeld oder ein Bedienfeld mit einer anderen GUI als AXIS wünschen.

- *-Wn panelname* - veranlasst HAL zu warten, bis die Komponente *panelname* fertig geladen ist (in der HAL-Sprache *bereit* werden), bevor weitere HAL-Befehle verarbeitet werden. Dies ist wichtig, da PyVCP-Panels HAL-Pins exportieren und andere HAL-Komponenten diese benötigen, um sich mit ihnen zu verbinden. Beachten Sie das große W und das kleine n. Wenn Sie die Option -Wn verwenden, müssen Sie die Option -c verwenden, um das Panel zu benennen.
- *pyvcp* < *-g*> < *-c*> *panel.xml* - erstellt das Panel mit der optionalen Geometrie und/oder dem Panel-Namen aus der Panel-XML-Datei. Die Datei panel.xml kann jeder Name sein, der auf .xml endet. Die XML-Datei ist die Datei, die beschreibt, wie das Panel zu erstellen ist. Sie müssen den Pfadnamen hinzufügen, wenn das Panel nicht in dem Verzeichnis liegt, in dem sich das HAL-Skript befindet.
- *-g <WxH><+X+Y>* - gibt die Geometrie an, die bei der Konstruktion des Panels verwendet werden soll. Die Syntax lautet *Breite x Höhe + X-Anker*
  Y-Anker. Sie können die Größe oder die Position oder beides festlegen. Der Ankerpunkt ist die obere linke Ecke des Panels. Ein Beispiel ist -g 250x500+800+0. Damit wird das Panel auf 250 Pixel Breite

und 500 Pixel Höhe eingestellt und bei X800 Y0 verankert.

- *-c panelname* - teilt PyVCP mit, wie die Komponente genannt werden soll und auch den Titel des Fensters. Der Panelname kann ein beliebiger Name ohne Leerzeichen sein.

Um ein *eigenständiges* PyVCP-Panel ohne LinuxCNC zu laden, verwenden Sie diesen Befehl:

```
loadusr -Wn mypanel pyvcp -g 250x500+800+0 -c mypanel mypanel.xml
```

Der minimale Befehl zum Laden eines PyVCP-Panels lautet:

```
loadusr pyvcp mypanel.xml
```

Sie würden diese verwenden, wenn Sie ein Panel ohne LinuxCNC's Maschine Controller wie für die Prüfung oder ein Standalone-DRO wollen.

Der Befehl loadusr wird verwendet, wenn Sie auch eine Komponente laden, die HAL am Schließen hindert, bis sie fertig ist. Wenn Sie ein Panel laden und dann Classic Ladder mit *loadusr -w classicladder* laden, würde CL HAL (und das Panel) offen halten, bis Sie CL schließen. Das obige *-Wn* bedeutet, dass Sie warten, bis die Komponente *-Wn "name"* bereit ist. (*name* kann ein beliebiger Name sein. Beachte das große W und das kleine n.) Das -c weist PyVCP an, ein Panel mit dem Namen *panel_file_name* unter Verwendung der Informationen in *panel_file_name.xml* zu erstellen. Der Name *panel_file_name.xml* kann ein beliebiger Name sein, muss aber auf .xml enden - es ist die Datei, die beschreibt, wie das Panel zu erstellen ist. Sie müssen den Pfadnamen hinzufügen, wenn das Panel nicht in dem Verzeichnis liegt, in dem sich das HAL-Skript befindet.

Ein optionaler Befehl, den Sie verwenden können, wenn Sie möchten, dass das Panel HAL daran hindert, Befehle fortzusetzen / herunterzufahren. Nach dem Laden anderer Komponenten soll dies der letzte HAL-Befehl sein:

```
waitusr panelname
```

Hiermit wird HAL angewiesen, auf das Schließen der Komponente *panelname* zu warten, bevor HAL-Befehle fortgesetzt werden. Dies wird in der Regel als letzter Befehl eingestellt, so dass HAL heruntergefahren wird, wenn das Panel geschlossen ist.

12.1.6. Widgets

HAL signals come in two variants, bools and numbers. Bools are off/on signals. Numbers can be *real*, *sint* or *uint*. For more information on HAL data types see the HAL Data section. The PyVCP widget can either display the value of the signal with an indicator widget, or modify the signal value with a control widget. Thus there are four classes of PyVCP widgets that you can connect to a HAL signal. A fifth class of helper widgets allow you to organize and label your panel.

- Widgets for indicating *bool* signals: `led`, `rectled`.
- Widgets for controlling *bool* signals: `button`, `checkbutton`, `radiobutton`.
- Widgets for indicating *number* signals: `number`, `sint`, `uint`, `bar`, `meter`.

- Widgets zur Steuerung von numerischen Signalen: `spinbox`, `scale`, `jogwheel`.
- Hilfs-Widgets: `hbox`, `vbox`, `table`, `label`, `labelframe`.

Syntax

Jedes Widget wird kurz beschrieben, gefolgt von dem verwendeten Markup und einem Screenshot. Alle Tags innerhalb des Haupt-Widget-Tags sind optional.

Allgemeine Anmerkungen

Gegenwärtig werden sowohl eine tag-basierte als auch eine attributbasierte Syntax unterstützt. So werden beispielsweise die folgenden XML-Fragmente identisch behandelt:

```
<led halpin="my-led"/>
```

und

```
<led><halpin>"my-led"</halpin></led>
```

Wenn die attributbasierte Syntax verwendet wird, werden die folgenden Regeln verwendet, um den Attributwert in einen Python-Wert zu verwandeln:

1. Wenn das erste Zeichen des Attributs eines der folgenden ist, wird es als Python-Ausdruck ausgewertet: `{(["'` .
2. Wird die Zeichenkette von int() akzeptiert, dann wird der Wert als Ganzzahl behandelt.
3. Bei Akzeptanz der Zeichenkette durch float() wird der Wert als Fließkommawert behandelt.
4. Andernfalls wird die Zeichenkette als Zeichenkette akzeptiert.

Bei Verwendung der Tag-basierten Syntax wird der Text innerhalb des Tags immer als Python-Ausdruck ausgewertet.

Die folgenden Beispiele zeigen eine Mischung aus verschiedenen Formaten.

Kommentare

Um einen Kommentar hinzuzufügen, verwenden Sie die XML-Syntax für einen Kommentar.

```
<!-- Mein Kommentar -->
```

Bearbeitung der XML-Datei

Bearbeiten Sie die XML-Datei mit einem Texteditor. In den meisten Fällen können Sie mit der rechten Maustaste auf die Datei klicken und "Mit Texteditor öffnen" oder ähnliches wählen.

Farben

Farben können unter Verwendung der X11-RGB-Farben mit dem Namen *gray75* oder hex *#0000ff* angegeben werden. Eine vollständige Liste befindet sich auf https://sedition.com/perl/rgb.html.

Gebräuchliche Farben (Nummern bezeichnen Schattierungen der jeweiligen Farbe)

- white (engl. für weiß)
- black (engl. für schwarz)
- blue (engl. für blau) und blue1 - 4
- cyan und cyan1 - 4
- green (engl. für grün) und green1 - 4
- yellow (engl. für gelb) und yellow1 - 4
- red (engl. für rot) und red1 - 4
- purple (engl. für violett) und purple1 - 4
- gray (amerikanisch für grau) und gray0 - 100

HAL-Pins

HAL-Pins bieten die Möglichkeit, das Widget mit etwas zu "verbinden". Sobald Sie einen HAL-Pin für Ihr Widget erstellt haben, können Sie ihn mit einem *net*-Befehl in einer .hal-Datei mit einem anderen HAL-Pin *verbinden*. Für weitere Informationen über den *net*-Befehl siehe den Abschnitt HAL Befehle.

Label

Ein Etikett ist eine Möglichkeit, Ihrem Panel Text hinzuzufügen.

- <*label*></*label*> - erstellt ein Label.
- <*text*>"*text*"</*text*> - der Text, der in das Etikett eingefügt werden soll; ein leeres Etikett kann als Abstandshalter verwendet werden, um andere Objekte auszurichten.
- <*font*>("*Helvetica*",*20*)</*font*> - Schriftart und -größe des Textes angeben.
- <*relief*>*FLAT*</*relief*> - Angabe des Rahmens um das Etikett (*FLAT*, *RAISED*, *SUNKEN*) Standard ist *FLAT*.
- <*bd*>*n*</*bd*> - wobei *n* die Breite des Rahmens ist, wenn *RAISED* oder *SUNKEN* verwendet wird.
- <*padx*>*n*</*padx*> - wobei *n* die Menge des zusätzlichen horizontalen Raums ist.
- <*pady*>*n*</*pady*> - wobei *n* für die Anzahl der zusätzlichen vertikalen Leerzeichen steht.

Das Etikett hat einen optionalen Deaktivierungsstift, der erstellt wird, wenn Sie <*disable_pin*>*True*</*disable_pin*> hinzufügen.

```
<label>
    <text>"Dies ist ein Label:"</text>
    <font>("Helvetica",20)</font>
</label>
```

Der obige Code ergab dieses Beispiel:



Figure 223. Beispiel für ein einfaches Etikett

Multi_Label

Eine Erweiterung der Textbeschriftung.

Selectable text label, can display up to 6 label legends when associated *bool* pin is activated.

Verbinden Sie jeden Legenden-Pin mit einem Signal und erhalten Sie eine beschreibende Bezeichnung, wenn das Signal TRUE ist.

Wenn mehr als ein Legenden-Pin TRUE ist, wird die Legende mit der höchsten Nummer "TRUE" angezeigt.

Wenn ein Deaktivierungs-Pin mit *<disable_pin>True</disable_pin>* erstellt wird und dieser Pin auf true gesetzt wird, ändert sich die Beschriftung in einen ausgegrauten Zustand.

```
<multilabel>
    <legends>["Label1", "Label2", "Label3", "Label4", "Label5", "Label6"]</legends>
    <font>("Helvetica",20)</font>
    <disable_pin>True</disable_pin>
</multilabel>
```

Das obige Beispiel würde die folgenden Pins erzeugen.

```
pyvcp.multilabel.0.disable
pyvcp.multilabel.0.legend0
pyvcp.multilabel.0.legend1
pyvcp.multilabel.0.legend2
pyvcp.multilabel.0.legend3
pyvcp.multilabel.0.legend4
pyvcp.multilabel.0.legend5
```

Wenn Sie mehr als ein Multilabel haben, würden die erzeugten Pins die Nummer wie folgt erhöhen: *pyvcp.multilabel.1.legend1*.

LEDs

A LED is used to indicate the status of a *bool* halpin. The LED color will be on_color when the halpin is true, and off_color otherwise.

- *<led></led>* - erzeugt eine runde LED
- *<rectled></rectled>* - erzeugt eine rechteckige LED
- *<halpin>name</halpin>* - Name des Pins, Standard ist *led.n*, wobei *n* eine ganze Zahl ist, die für jede LED inkrementiert wird.

- <*size*>*n*</*size*> - *n* ist die Größe der LED in Pixeln, Standardwert ist 20.
- <on_color>*farbe*</on_color>' - legt die Farbe der LED fest auf *farbe*, wenn der Pin wahr ist. Voreinstellung ist *green* (engl. für grün). Siehe Abschnitt zu Farben für weitere Informationen.
- <*off_color*>*farbe*</*off_color*> - legt die Farbe der LED fest auf *farbe*, wenn der Pin falsch ist. Voreinstellung ist *red* (engl. für rot).
- <*height*>*n*</*height*> - legt die Höhe der LED in Pixeln fest.
- <*width*>*n*</*width*> - legt die Breite der LED in Pixeln fest.
- <*disable_pin*>*false*</*disable_pin*> - bei true wird der LED ein Deaktivierungspin hinzugefügt.
- <*disabled_color*>*color*</*disabled_color*> - setzt die Farbe der LED auf *color*, wenn der Pin deaktiviert ist.

Runde LED

```
<led>
    <halpin>"meine-LED"</halpin>
    <size>50</size>
    <on_color>"green"</on_color>
    <off_color>"red"</off_color>
</led>
```

Der obige Code ergab dieses Beispiel:

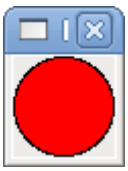


Figure 224. Beispiel für eine runde LED

Rechteckige LED

Dies ist eine Variante des "led" -Widgets.

```
<vbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <rectled>
        <halpin>"meine-led"</halpin>
        <height>"50"</height>
        <width>"100"</width>
        <on_color>"green"</on_color>
        <off_color>"red"</off_color>
    </rectled>
</vbox>
```

Der obige Code erzeugt dieses Beispiel. Es zeigt auch einen vertikalen Kasten mit Relief.



Figure 225. Beispiel für eine einfache Rechteck-LED

Buttons

A button is used to control a BOOL pin. The pin will be set True when the button is pressed and held down, and will be set False when the button is released. Buttons can use the following optional options.

- *<padx>n</padx>* - wobei *n* die Menge des zusätzlichen horizontalen Raums ist.
- *<pady>n</pady>* - wobei *n* für die Anzahl der zusätzlichen vertikalen Leerzeichen steht.
- *<activebackground>"color"</activebackground>* - der Cursor über Farbe auf *color* gesetzt.
- *<fg>"color"</fg>* - die Vordergrundfarbe wird auf *color* gesetzt.
- *<bg>"color"</bg>* - die auf *color* gesetzte Hintergrundfarbe.
- *<disable_pin>True</disable_pin>* - Pin deaktivieren.

Text Button

A text button controls a *bool* halpin. The halpin is false until the button is pressed then it is true. The button is a momentary button.

Die Text Button hat einen optionalen Deaktivierungsstift, der beim Hinzufügen von <disable_pin>True</disable_pin> angelegt wird.

```
<button>
    <halpin>"ok-button"</halpin>
    <text>"OK"</text>
</button>
<button>
    <halpin>"abort-button"</halpin>
    <text>"Abort"</text>
</button>
```

Der obige Code ergab dieses Beispiel:



Figure 226. Beispiel für einfache Buttons

Checkbutton

A checkbutton controls a *bool* halpin. The halpin will be set True when the button is checked, and false when the button is unchecked. The checkbutton is a toggle type button. The checkbuttons may be set

initially as TRUE or FALSE the initval field A pin called changepin is also created automatically, which can toggle the Checkbutton via HAL, if the value linked is changed, to update the display remotely.



Figure 227. Nicht-markierter (engl. unchecked) button



Figure 228. Markierter (engl. checked) Button

Checkbutton Code Beispiel

```
<checkbutton>
    <halpin>"coolant-chkbtn"</halpin>
    <text>"Coolant"</text>
    <initval>1</initval>
</checkbutton>
<checkbutton>
    <halpin>"chip-chkbtn"</halpin>
    <text>"Chips    "</text>
    <initval>0</initval>
</checkbutton>
```

Der obige Code ergab dieses Beispiel:

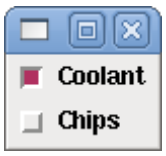


Figure 229. Einfaches Checkbutton-Beispiel

Das Kontrollkästchen für das Kühlmittel ist aktiviert.
Beachten Sie die zusätzlichen Leerzeichen im Text "Chips", um die Kontrollkästchen auszurichten.

Radiobutton

Ein Radiobutton setzt einen der HAL Pins auf true. Die anderen Pins werden auf false gesetzt. Das Feld initval kann eingestellt werden, um die Standardauswahl zu treffen, wenn das Panel angezeigt wird. Es darf nur ein Radiobutton auf TRUE (1) gesetzt werden, oder nur der Pin mit der höchsten Nummer, der auf TRUE gesetzt wurde, erhält diesen Wert.

```
<radiobutton>
    <choices>["one","two","three"]</choices>
    <halpin>"my-radio"</halpin>
    <initval>0</initval>
</radiobutton>
```

Der obige Code ergab dieses Beispiel:

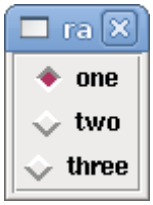


Figure 230. Einfaches Radiobutton-Beispiel

Beachten Sie, dass die HAL-Pins im obigen Beispiel my-radio.one, my-radio.two und my-radio.three heißen werden. In der obigen Abbildung ist *eins* (engl. one) der ausgewählte Wert. Verwenden Sie das Tag "<orient>HORIZONTAL</orient>" für die horizontale Anzeige.

Nummernanzeigen

Für die Anzeige von Zahlen stehen folgende Formatierungsoptionen zur Verfügung

- `<font>("Font Name",n)</font>` wobei *n* die Schriftgröße ist.
- `<width>_n_</width>`, wobei *n* die Gesamtbreite des verwendeten Raums ist.
- `<justify>_pos_</justify>`, wobei *pos* LEFT, CENTER oder RIGHT ist (funktioniert nicht).
- `<padx>__n__</padx>`, wobei *n* die Menge des zusätzlichen horizontalen Raums ist.
- `<pady>__n__</pady>`, - wobei *n* für die Anzahl der zusätzlichen vertikalen Leerzeichen steht.

Nummer

The number widget displays the value of a real signal.

```
<number>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"+4.4f"</format>
</number>
```

Der obige Code ergab dieses Beispiel:

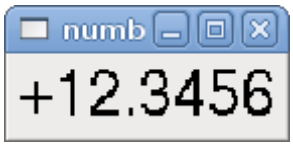


Figure 231. Beispiel für einfache Zahlen

- *<font>* - ist eine Tkinter-Schriftart und Größenangabe. Eine Schriftart, die mindestens bis zur Größe 200 angezeigt wird, ist "courier 10 pitch", so dass Sie für ein wirklich großes Zahlen-Widget angeben können:

```
<font>("courier 10 pitch",100)</font>
```

- *<format>* - ist ein angegebenes Format im C-Stil, das bestimmt, wie die Zahl angezeigt wird.

sint Number

The *sint* number widget displays the value of a `sint` number. The syntax is the same as *number* except the name which is `<sint>`. Make sure the width is wide enough to cover the largest number you expect to use. This was formerly known as `<s32>`, which is now an alias for `<sint>`.

```
<sint>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"6d"</format>
    <width>6</width>
</sint>
```

Der obige Code ergab dieses Beispiel:

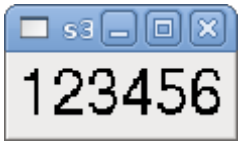


Figure 232. Simple sint Number Example

uint Number

The *uint* number widget displays the value of a `uint` number. The syntax is the same as *number* except the name which is `<uint>`. This was formerly known as `<u32>`, which is now an alias for `<uint>`.

Bar (engl. für Balken)

A bar widget displays the value of a *real* signal both graphically using a bar display and numerically. The color of the bar can be set as one color throughout its range (default using fillcolor) or set to change color, dependent upon the value of the halpin (range1, range2 range3 must all be set, if you only want 2 ranges, set 2 of them to the same color).

- `<halpin>"my-bar"</halpin>` text, setzt den Pin-Namen: `pyvcp.my-bar`.
- `<min_>0</min_>` Zahl, legt die minimale Skalierung fest.
- `<max_>140</max_>` (Zahl), legt die maximale Skalierung fest.
- `<format>"3.1f"</format>` (Text), setzt das Zahlenformat mit Python-Zahlenformatierung.
- `<bgcolor>"grau"</bgcolor>` (Text), legt die Hintergrundfarbe fest.
- `<fillcolor>"rot"</fillcolor>` (Text), setzt die Füllfarbe.
- `<range1>0,100, "grün"</range1>` (Zahl, Zahl, Text), legt den ersten Bereich und die Farbe fest.
- `<range2>101,135, "orange"</range2>` (number, number, text), legt den ersten Bereich und die Farbe fest.
- <range3>136, 150, "rot"</range3> Zahl, Zahl, Text, legt den ersten Bereich und die Farbe fest.
- `<canvas_width>200</canvas_width>`, (Zahl), legt die Gesamtbreite fest.
- `<canvas_height>50</canvas_height>` (Zahl), legt die Gesamthöhe fest.
- `<bar_height>30</bar_height>` (Zahl), legt die Balkenhöhe fest, muss kleiner sein als canvas_height.
- `<bar_width>150</bar_width>` (Zahl), setzt die Balkenbreite, muss kleiner als canvas_width sein.

```
<bar>
    <halpin>"my-bar"</halpin>
    <min_>0</min_>
    <max_>123</max_>
    <format>"3.1f"</format>
    <bgcolor>"grey"</bgcolor>
    <fillcolor>"red"</fillcolor>
    <range1>0,100,"green"</range1>
    <range2>101,135,"orange"</range2>
    <range3>136, 150,"red"</range3>
    <canvas_width>200</canvas_width>
    <canvas_height>50</canvas_height>
    <bar_height>30</bar_height>
    <bar_width>150</bar_width>
</bar>
```

Der obige Code ergab dieses Beispiel:

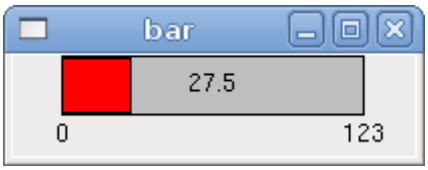


Figure 233. Einfaches Bar-Beispiel

Meter

Meter displays the value of a *real* signal using a traditional dial indicator.

```
<meter>
    <halpin>"mymeter"</halpin>
    <text>"Battery"</text>
    <subtext>"Volts"</subtext>
    <size>250</size>
    <min_>0</min_>
    <max_>15.5</max_>
    <majorscale>1</majorscale>
    <minorscale>0.2</minorscale>
    <region1>(14.5,15.5,"yellow")</region1>
    <region2>(12,14.5,"green")</region2>
    <region3>(0,12,"red")</region3>
</meter>
```

Der obige Code ergab dieses Beispiel:



Figure 234. Einfaches Beispiel für ein Messgerät

Numerische Eingaben

Spinbox (Einstellrad)

A spinbox controls a *real* pin. You increase or decrease the value of the pin by either pressing on the arrows, or pointing at the spinbox and rolling your mouse-wheel. If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input. If the reset_button field is set TRUE(1), a button is placed below the spinbox which sets the value back to the one given by initval. The reset_text field sets the text shown on that button and defaults to "reset".

```
<spinbox>
    <halpin>"my-spinbox"</halpin>
    <min_>-12</min_>
    <max_>33</max_>
    <initval>0</initval>
    <resolution>0.1</resolution>
    <format>"2.3f"</format>
    <font>("Arial",30)</font>
    <param_pin>1</param_pin>
</spinbox>
```

Der obige Code ergab dieses Beispiel:



Figure 235. Einfaches Spinbox-Beispiel

A spinbox with a reset button, which puts the value back to zero with one click:

```
<spinbox>
    <halpin>"dac-setpoint"</halpin>
```

```
    <min_>-10</min_>
    <max_>10</max_>
    <initval>0</initval>
    <resolution>0.1</resolution>
    <format>"+03.1f"</format>
    <reset_button>1</reset_button>
    <reset_text>"back to zero"</reset_text>
</spinbox>
```

Skala

A scale controls a `real` or a `sint` pin. You increase or decrease the value of the pin be either dragging the slider, or pointing at the scale and rolling your mouse-wheel. The *halpin* will have both *-f* and *-i* added to it to form the `real` and `sint` pins. Width is the width of the slider in vertical and the height of the slider in horizontal orientation. If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input.

```
<scale>
    <font>("Helvetica",16)</font>
    <width>"25"</width>
    <halpin>"my-hscale"</halpin>
    <resolution>0.1</resolution>
    <orient>HORIZONTAL</orient>
    <initval>-15</initval>
    <min_>-33</min_>
    <max_>26</max_>
    <param_pin>1</param_pin>
</scale>
<scale>
    <font>("Helvetica",16)</font>
    <width>"50"</width>
    <halpin>"my-vscale"</halpin>
    <resolution>1</resolution>
    <orient>VERTICAL</orient>
    <min_>100</min_>
    <max_>0</max_>
    <param_pin>1</param_pin>
</scale>
```

Der obige Code ergab dieses Beispiel:

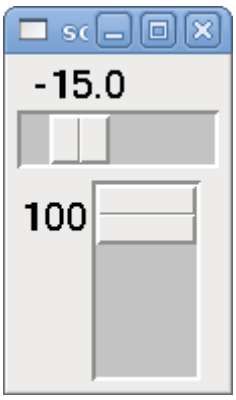


Figure 236. Beispiel für eine einfache Skalierung

NOTE Beachten Sie, dass standardmäßig "min" angezeigt wird, auch wenn es größer als "max" ist, es sei denn, "min" ist negativ.

Wählscheibe (engl. dial)

The Dial outputs a HAL real and reacts to both mouse wheel and dragging. Double left click to increase the resolution and double right click to reduce the resolution by one digit. The output is capped by the min and max values. The <cpr> is how many tick marks are on the outside of the ring (beware of high numbers). If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input.

```
<dial>
    <size>200</size>
    <cpr>100</cpr>
    <min_>-15</min_>
    <max_>15</max_>
    <text>"Dial"</text>
    <initval>0</initval>
    <resolution>0.001</resolution>
    <halpin>"anaout"</halpin>
    <dialcolor>"yellow"</dialcolor>
    <edgecolor>"green"</edgecolor>
    <dotcolor>"black"</dotcolor>
    <param_pin>1</param_pin>
</dial>
```

Der obige Code ergab dieses Beispiel:

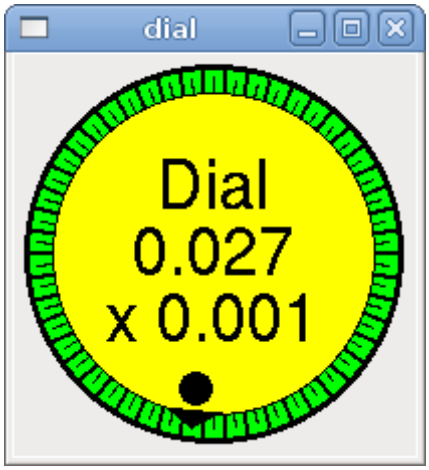


Figure 237. Einfaches Wählbeispiel

Jogwheel

Jogwheel mimics a real jogwheel by outputting a *real* pin which counts up or down as the wheel is turned, either by dragging in a circular motion, or by rolling the mouse-wheel.

Optionale Tags:

- *<text>"Mein Text"</text>* zeigt Text an
- *<bgcolor>"grey"</bgcolor> <fillcolor>"green"</fillcolor>* Hintergrund- und aktive Farben

- *<scale_pin>1</scale_pin>* creates scale text and a `REAL.scale` pin to display jog scale
- *<clear_pin>1</clear_pin>* creates `DRO` and a `BOOL.reset` pin to reset DRO. Needs scale_pin for scaled DRO. Shift+click resets DRO also

```
<jogwheel>
    <halpin>"my-wheel"</halpin>
    <cpr>45</cpr>
    <size>250</size>
</jogwheel>
```

Der obige Code ergab dieses Beispiel:

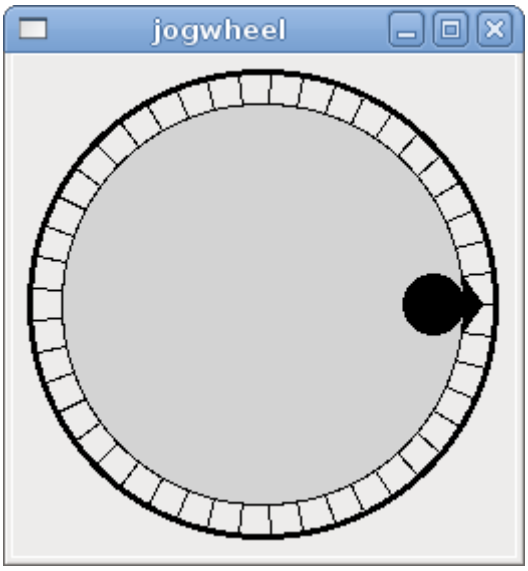


Figure 238. Einfaches Jogwheel-Beispiel

Bilder

Die Bilder dürfen nur im .gif-Format angezeigt werden. Alle Bilder müssen die gleiche Größe haben. Die Bilder müssen sich im gleichen Verzeichnis wie Ihre INI-Datei befinden (oder im aktuellen Verzeichnis, wenn Sie das Programm von der Kommandozeile aus mit halrun/halcmd ausführen).

Image Bit

The *image_bool* toggles between two images by setting the halpin to true or false. This was formerly known as *image_bit*, which is now an alias for *image_bool*.

```
<image name='fwd' file='fwd.gif'/>
<image name='rev' file='rev.gif'/>
<vbox>
    <image_bool halpin='selectimage' images='fwd rev'/>
</vbox>
```

Dieses Beispiel wurde mit dem obigen Code erstellt. Es verwendet die beiden Bilddateien fwd.gif und rev.gif. FWD wird angezeigt, wenn "selectimage" falsch ist, und REV wird angezeigt, wenn "selectimage" wahr ist.



Figure 239. Selectimage False Beispiel



Figure 240. Selectimage True Beispiel

Image uint

The *image_uint* is the same as *image_bool*, except you have essentially an unlimited number of images and you *select* the image by setting the halpin to a integer value with 0 for the first image in the images list and 1 for the second image, etc. This was formerly known as *image_u32*, which is now an alias for *image_uint*.

```
<image name='stb' file='stb.gif'/>
<image name='fwd' file='fwd.gif'/>
<image name='rev' file='rev.gif'/>
<vbox>
    <image_uint halpin='selectimage' images='stb fwd rev'/>
</vbox>
```

Der obige Code erzeugt das folgende Beispiel, indem das Bild stb.gif hinzugefügt wird.



Figure 241. Simple image_uint Example with halpin=0



Figure 242. Simple image_uint Example withhalpin=1



Figure 243. Simple image_uint Example withhalpin=2

Beachten Sie, dass der Standardwert der Minimalwert ist, auch wenn er höher als der Maximalwert ist, es sei denn, es gibt einen negativen Minimalwert.

Containers (engl. für Behälter)

Container sind Widgets, die andere Widgets enthalten. Container werden verwendet, um andere Widgets zu gruppieren.

Begrenzungen (engl. borders)

Containerränder werden mit zwei Tags angegeben, die zusammen verwendet werden. Der <relief>-Tag gibt die Art des Rahmens an, und der <bd>-Tag gibt die Breite des Rahmens an.

<relief>_Typ_</relief>

Wobei *Typ* FLAT, SUNKEN, RAISED, GROOVE, oder RIDGE ist.

<bd>_n_</bd>

Dabei ist *n* die Breite des Rahmens.

```
<hbox>
    <button>
        <relief>FLAT</relief>
        <text>"FLAT"</text>
        <bd>3</bd>
    </button>
    <button>
        <relief>SUNKEN</relief>
        <text>"SUNKEN"</text>
        <bd>3</bd>
    </button>
    <button>
        <relief>RAISED</relief>
        <text>"RAISED"</text>
        <bd>3</bd>
    </button>
    <button>
        <relief>GROOVE</relief>
        <text>"GROOVE"</text>
        <bd>3</bd>
    </button>
    <button>
        <relief>RIDGE</relief>
        <text>"RIDGE"</text>
        <bd>3</bd>
    </button>
```

```
</hbox>
```

Der obige Code ergab dieses Beispiel:



Figure 244. Container Borders Showcase

Ausfüllen (engl. fill)

Containerfüllungen werden mit dem Tag <*boxfill fill=""*/> angegeben. Gültige Einträge sind none, x, y und both. Die x-Füllung ist eine horizontale Füllung und die y-Füllung ist eine vertikale Füllung

<boxfill fill ="style"/>

Dabei ist *style* none, x, y oder beides. Standardwert ist x für Vbox und y für Hbox.

Anker

Container-Anker werden mit dem Tag <boxanchor anchor=""/> angegeben. Der Anker gibt an, wo jeder Slave in seiner Parzelle positioniert werden soll. Gültige Einträge sind center, n, s, e, w, für center, north, south, east und west. Kombinationen wie sw, se, nw und ne sind ebenfalls gültig.

<boxanchor anchor="position"/>

Dabei ist *Position* center, n, s, e, w, ne, nw, se oder sw. Standardwert ist Mitte.

Erweitern

Container expand wird mit dem booleschen Tag <boxexpand expand=""/> angegeben. Gültige Einträge sind "yes", "no".

<boxexpand expand="boolean"/>

Wobei *boolean* entweder "yes" (engl. für ja) oder "no" (nein) ist. Die Voreinstellung ist "yes".

Hbox

Verwenden Sie eine Hbox, wenn Sie Widgets horizontal nebeneinander stapeln möchten.

```
<hbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>"a hbox:"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</hbox>
```

Der obige Code ergab dieses Beispiel:

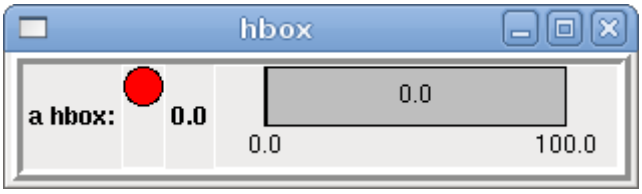


Figure 245. Einfaches hbox-Beispiel

Innerhalb einer Hbox können Sie die Tags <*boxfill fill=""/*>, <*boxanchor anchor=""/*> und <*boxexpand expand=""/*> verwenden, um festzulegen, wie sich die Elemente in der Box verhalten, wenn die Größe des Fensters geändert wird.Die Standardeinstellung ist *fill="y"*, *anchor="center"*, *expand="yes"* für eine Hbox.

Vbox

Verwenden Sie eine Vbox, wenn Sie Widgets vertikal übereinander stapeln möchten.

```
<vbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>"eine vbox:"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</vbox>
```

Der obige Code ergab dieses Beispiel:

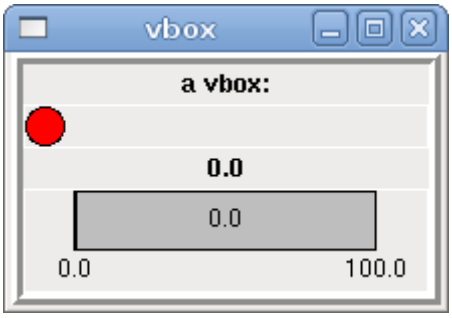


Figure 246. Einfaches vbox-Beispiel

Innerhalb einer Vbox können Sie die Tags <*boxfill fill=""/*>, <*boxanchor anchor=""/*>, jnd <*boxexpand expand=""/*> verwenden, um festzulegen, wie sich die Elemente in der Box verhalten, wenn die Größe des Fensters geändert wird. Die Standardeinstellung ist *fill="x"*, *anchor="center"*, *expand="yes"* für eine Vbox.

Etikettrahmen (engl. labelframe)

Ein Etikettenrahmen (engl. labelframe) ist ein Rahmen mit einer Rille und einem Etikett in der oberen linken Ecke.

```
<labelframe text="Label: Leds groupées">
```

```
<labelframe text="Group Title">
    <font>("Helvetica",16)</font>
```

```
    <hbox>
    <led/>
    <led/>
    </hbox>
</labelframe>
```

Der obige Code ergab dieses Beispiel:



Figure 247. Einfaches Labelframe-Beispiel

Tabelle (engl. table)

Eine Tabelle (engl. table) ist ein Container, der das Layout in einem Raster aus Zeilen und Spalten ermöglicht. Jede Zeile wird mit einem "<tablerow/>"-Tag eingeleitet. Ein enthaltenes Widget kann sich über Zeilen oder Spalten erstrecken, indem es das Tag "<tablespan rows= cols=/>" verwendet. Die Seiten der Zellen, an denen die enthaltenen Widgets "kleben", können durch die Verwendung des Tags "<tablesticky sticky=/>" festgelegt werden. Eine Tabelle dehnt sich über ihre flexiblen Zeilen und Spalten aus.

Tabelle Code Beispiel

```
<table flexible_rows="[2]" flexible_columns="[1,4]">
<tablesticky sticky="new"/>
<tablerow/>
    <label>
        <text>" A (cell 1,1) "</text>
        <relief>RIDGE</relief>
        <bd>3</bd>
    </label>
    <label text="B (cell 1,2)"/>
    <tablespan columns="2"/>
    <label text="C, D (cells 1,3 and 1,4)"/>
<tablerow/>
    <label text="E (cell 2,1)"/>
    <tablesticky sticky="nsew"/>
    <tablespan rows="2"/>
    <label text="'spans\n2 rows'"/>
    <tablesticky sticky="new"/>
    <label text="G (cell 2,3)"/>
    <label text="H (cell 2,4)"/>
<tablerow/>
    <label text="J (cell 3,1)"/>
    <label text="K (cell 3,2)"/>
    <uint halpin="test"/>
</table>
```

Der obige Code ergab dieses Beispiel:

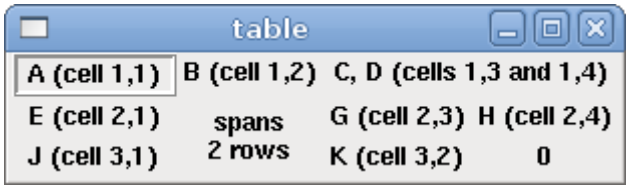


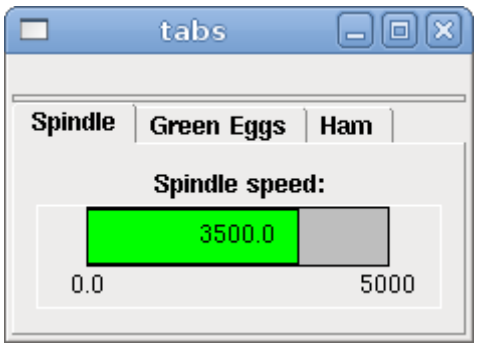
Figure 248. Beispiel für eine Tabelle

Registerkarten (engl. tabs)

Eine Benutzeroberfläche mit Registerkarten kann ziemlich viel Platz sparen.

```
<tabs>
    <names> ["spindle","green eggs"]</names>
</tabs>
<tabs>
    <names>["Spindle", "Green Eggs", "Ham"]</names>
    <vbox>
        <label>
            <text>"Spindle speed:"</text>
        </label>
        <bar>
            <halpin>"spindle-speed"</halpin>
            <max_>5000</max_>
        </bar>
    </vbox>
    <vbox>
        <label>
            <text>"(this is the green eggs tab)"</text>
        </label>
    </vbox>
    <vbox>
        <label>
            <text>"(this tab has nothing on it)"</text>
        </label>
    </vbox>
</tabs>
```

Der obige Code ergibt dieses Beispiel, in dem jede ausgewählte Registerkarte angezeigt wird.



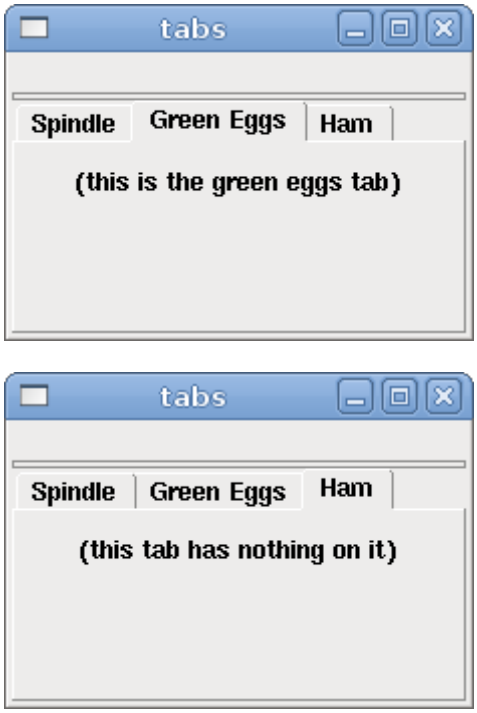


Figure 249. Einfache Tabs-Beispiel

12.2. PyVCP-Beispiele

12.2.1. Testing a Panel Without HAL

A PyVCP panel can be opened directly with the `pyvcp` command to check the layout and widgets before connecting it to HAL. Put the following in a file named `hello_world.xml`:

```
<pyvcp>
  <vbox>
    <label>
      <text>"Hello CNC"</text>
    </label>
  </vbox>
</pyvcp>
```

Then run the command from a terminal:

```
pyvcp hello_world.xml
```

This opens the panel as a standalone window. It makes the edit/test cycle fast because no HAL setup is required.

12.2.2. ACHSE

Um ein PyVCP-Panel zur Verwendung mit der AXIS-Schnittstelle zu erstellen, die rechts von AXIS angebracht ist, müssen Sie die folgenden grundlegenden Dinge tun.

- Erstellen Sie eine XML-Datei, die Ihre Panel-Beschreibung enthält, und legen Sie sie in Ihr

Konfigurationsverzeichnis.

- Fügen Sie den PyVCP-Eintrag in den [DISPLAY]-Abschnitt der INI-Datei mit dem Namen Ihrer XML-Datei ein.
- Fügen Sie den Eintrag POSTGUI_HALFILE in den [HAL]-Abschnitt der INI-Datei ein und geben Sie den Namen Ihrer postgui-HAL-Datei an.
- Fügen Sie die Links zu HAL-Pins für Ihr Panel in der Datei postgui.hal hinzu, um Ihr PyVCP-Panel mit LinuxCNC zu *verbinden*.

12.2.3. Schwebende (engl. floating) Panels

Um schwebende PyVCP-Panels zu erstellen, die mit jeder Schnittstelle verwendet werden können, müssen Sie die folgenden grundlegenden Dinge tun.

- Erstellen Sie eine XML-Datei, die Ihre Panel-Beschreibung enthält, und legen Sie sie in Ihr Konfigurationsverzeichnis.
- Fügen Sie eine loadusr-Zeile in Ihre HAL-Datei ein, um jedes Panel zu laden.
- Fügen Sie die Links zu HAL-Pins für Ihr Panel in der Datei postgui.hal hinzu, um Ihr PyVCP-Panel mit LinuxCNC zu *verbinden*.

Nachfolgend ein Beispiel für einen loadusr-Befehl, um zwei PyVCP-Panels zu laden und jedes zu benennen, damit die Verbindungsnamen in HAL bekannt sind.

```
loadusr -Wn btnpanel pyvcp -c btnpanel panel1.xml
loadusr -Wn sppanel pyvcp -c sppanel panel2.xml
```

Die Option -Wn bewirkt, dass HAL darauf wartet, dass der Name geladen wird, bevor es weitergeht.

Die Option pyvcp -c sorgt dafür, dass PyVCP das Panel benennt.

Die HAL-Pins aus panel1.xml werden als btnpanel.<_Pinname_> bezeichnet.

Die HAL-Pins aus panel2.xml werden als btnpanel.<_Pinname_> bezeichnet.

Stellen Sie sicher, dass die loadusr-Zeile vor allen Netzen steht, welche die PyVCP-Pins verwenden.

12.2.4. Beispiel für Jog-Buttons

In diesem Beispiel erstellen wir ein PyVCP-Panel mit Jog-Buttons für X, Y und Z. Diese Konfiguration wird auf einer vom StepConf-Assistenten generierten Konfiguration aufgebaut. Zunächst führen wir den StepConf-Assistenten aus und konfigurieren unseren Rechner. Auf der Seite Erweiterte Konfigurationsoptionen fügen wir dann ein leeres PyVCP-Panel hinzu, wie in der folgenden Abbildung gezeigt. Für dieses Beispiel haben wir die Konfiguration auf der Seite mit den grundlegenden Maschineninformationen des StepConf-Assistenten "pyvcp_xyz" genannt.

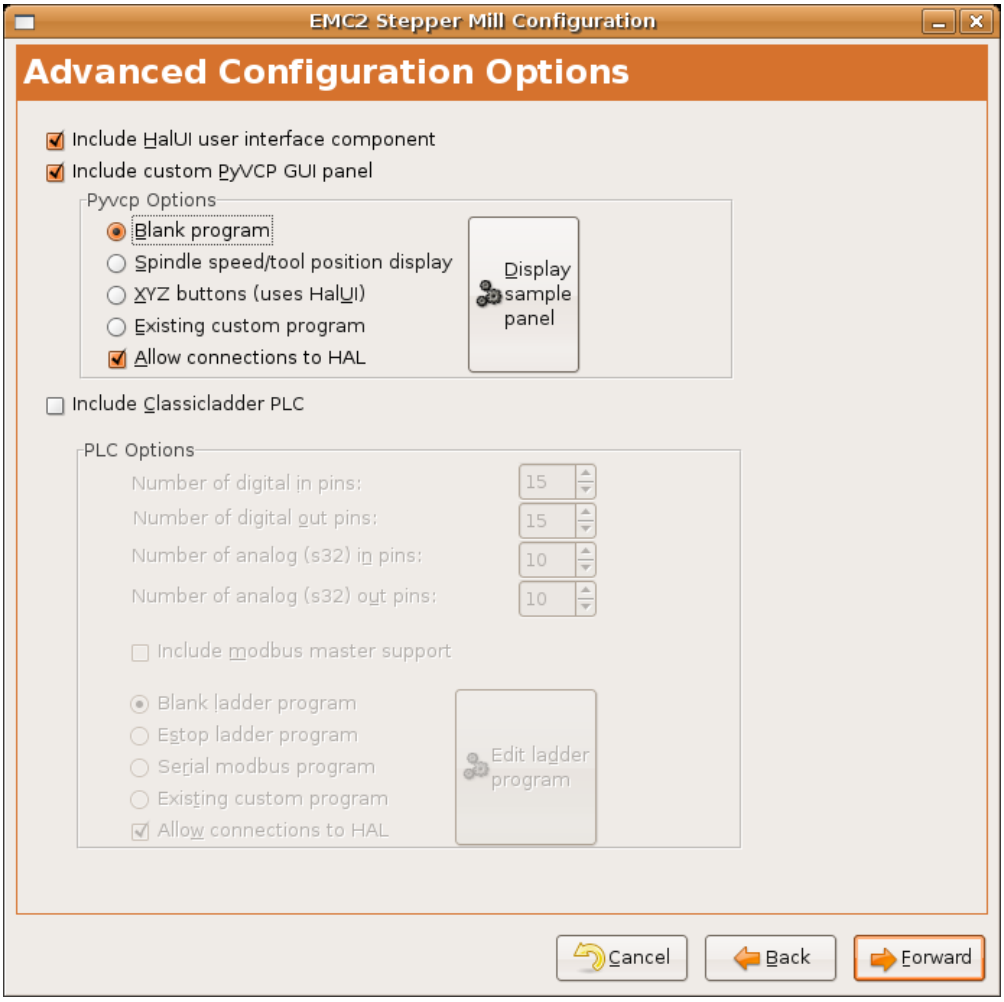


Figure 250. XYZ-Assistent Konfiguration

Der StepConf-Assistent erstellt mehrere Dateien und legt sie im Verzeichnis `linuxcnc/configs/pyvcp_xyz` ab. Wenn Sie die Option "Link erstellen" aktiviert haben, finden Sie auf Ihrem Desktop einen Link zu diesen Dateien.

Erstellen der Widgets

Öffnen Sie die Datei custompanel.xml, indem Sie mit der rechten Maustaste darauf klicken und "Mit Texteditor öffnen" wählen. Zwischen den Tags <pyvcp></pyvcp> fügen wir die Widgets für unser Panel ein.

Schauen Sie in den Abschnitt PyVCP Widgets Referenz des Handbuchs für detailliertere Informationen über jedes Widget documentation des widgets.

In der Datei custompanel.xml werden wir die Beschreibung der Widgets hinzufügen.

```
<pyvcp>
  <labelframe text="Jog Buttons">
    <font>("Helvetica",16)</font>

      <!-- the X jog buttons -->
      <hbox>
      <relief>RAISED</relief>
      <bd>3</bd>
```

```
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"x-plus"</halpin>
        <text>"X+"</text>
      </button>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"x-minus"</halpin>
        <text>"X-"</text>
      </button>
      </hbox>

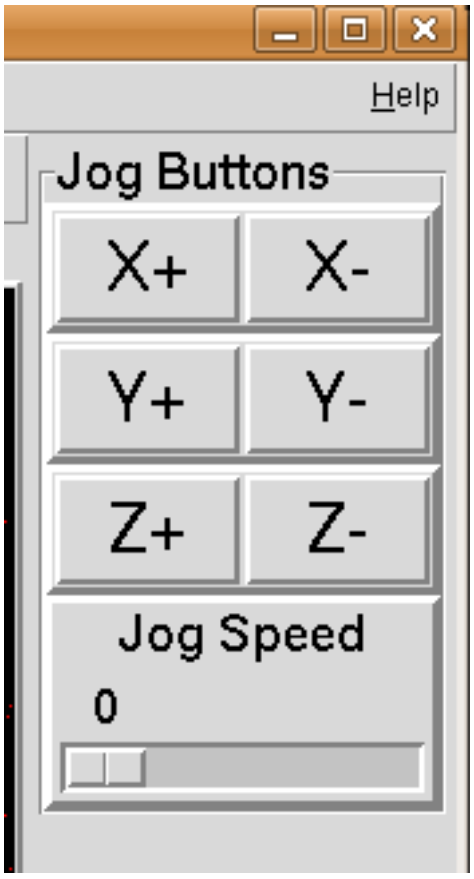
      <!-- the Y jog buttons -->
      <hbox>
      <relief>RAISED</relief>
      <bd>3</bd>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"y-plus"</halpin>
        <text>"Y+"</text>
      </button>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"y-minus"</halpin>
        <text>"Y-"</text>
      </button>
      </hbox>

      <!-- the Z jog buttons -->
      <hbox>
      <relief>RAISED</relief>
      <bd>3</bd>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"z-plus"</halpin>
        <text>"Z+"</text>
      </button>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"z-minus"</halpin>
        <text>"Z-"</text>
      </button>
      </hbox>

      <!-- the jog speed slider -->
      <vbox>
      <relief>RAISED</relief>
      <bd>3</bd>
      <label>
        <text>"Jog Speed"</text>
```

```
        <font>("Helvetica",16)</font>
      </label>
      <scale>
        <font>("Helvetica",14)</font>
        <halpin>"jog-speed"</halpin>
        <resolution>1</resolution>
        <orient>HORIZONTAL</orient>
        <min_>0</min_>
        <max_>80</max_>
      </scale>
      </vbox>
  </labelframe>
</pyvcp>
```

Nach dem Hinzufügen des oben genannten haben Sie nun ein PyVCP-Panel, das wie das folgende aussieht und an der rechten Seite von AXIS angebracht ist. Es sieht schön aus, aber es tut nichts, bis Sie die Tasten mit Halui "verbinden". Wenn Sie einen Fehler erhalten, wenn Sie versuchen, und führen Sie nach unten scrollen, um den unteren Rand des Pop-up-Fenster und in der Regel der Fehler ist ein Rechtschreib-oder Syntaxfehler und es wird dort sein.



Verbindungen herstellen

Um die erforderlichen Verbindungen herzustellen, öffnen Sie die Datei custom_postgui.hal, und fügen Sie Folgendes hinzu.

```
# Verbinden Sie die X-PyVCP-Tasten
net my-jogxminus halui.axis.x.minus <= pyvcp.x-minus
net my-jogxplus halui.axis.x.plus <= pyvcp.x-plus
```

```
# Verbinden der Y-PyVCP-Tasten
net my-jogyminus halui.axis.y.minus <= pyvcp.y-minus
net my-jogyplus halui.axis.y.plus <= pyvcp.y-plus

# Verbinden der Z-PyVCP-Tasten
net my-jogzminus halui.axis.z.minus <= pyvcp.z-minus
net my-jogzplus halui.axis.z.plus <= pyvcp.z-plus

# den PyVCP-Jog-Speed-Schieberegler anschließen
net my-jogspeed halui.axis.jog-speed <= pyvcp.jog-speed-f
```

Nachdem Sie den Notaus (engl. E-Stop) zurückgesetzt und in den Jog-Modus versetzt haben und den Schieberegler für die Jpggeschwindigkeit im PyVCP-Bedienfeld auf einen Wert größer als Null gestellt haben, sollten die PyVCP-Tipptasten funktionieren. Sie können nicht joggen, wenn eine G-Code-Datei ausgeführt wird, wenn das Programm pausiert oder wenn die Registerkarte MDI ausgewählt ist.

12.2.5. Port-Tester

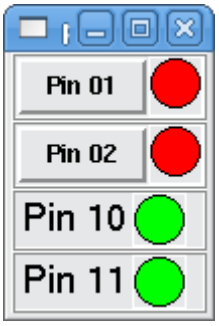
Dieses Beispiel zeigt Ihnen, wie Sie mit PyVCP und HAL einen einfachen Parallelport-Tester erstellen können.

Erstellen Sie zunächst die Datei ptest.xml mit dem folgenden Code, um die Beschreibung des Panels zu erstellen.

```
<!-- Test panel for the parallel port cfg for out -->
<pyvcp>
  <hbox>
    <relief>RIDGE</relief>
    <bd>2</bd>
    <button>
      <halpin>"btn01"</halpin>
      <text>"Pin 01"</text>
    </button>
    <led>
      <halpin>"led-01"</halpin>
      <size>25</size>
      <on_color>"green"</on_color>
      <off_color>"red"</off_color>
    </led>
  </hbox>
  <hbox>
    <relief>RIDGE</relief>
    <bd>2</bd>
    <button>
      <halpin>"btn02"</halpin>
      <text>"Pin 02"</text>
    </button>
    <led>
      <halpin>"led-02"</halpin>
      <size>25</size>
      <on_color>"green"</on_color>
      <off_color>"red"</off_color>
```

```
    </led>
  </hbox>
  <hbox>
    <relief>RIDGE</relief>
    <bd>2</bd>
    <label>
      <text>"Pin 10"</text>
      <font>("Helvetica",14)</font>
    </label>
    <led>
      <halpin>"led-10"</halpin>
      <size>25</size>
      <on_color>"green"</on_color>
      <off_color>"red"</off_color>
    </led>
  </hbox>
  <hbox>
    <relief>RIDGE</relief>
    <bd>2</bd>
    <label>
      <text>"Pin 11"</text>
      <font>("Helvetica",14)</font>
    </label>
    <led>
      <halpin>"led-11"</halpin>
      <size>25</size>
      <on_color>"green"</on_color>
      <off_color>"red"</off_color>
    </led>
  </hbox>
</pyvcp>
```

Dadurch wird das folgende schwebende Panel erstellt, das einige Eingangs- und Ausgangsanschlüsse enthält.



Um die HAL-Befehle auszuführen, die wir benötigen, um alles zum Laufen zu bringen, fügen wir Folgendes in unsere Datei ptest.hal ein.

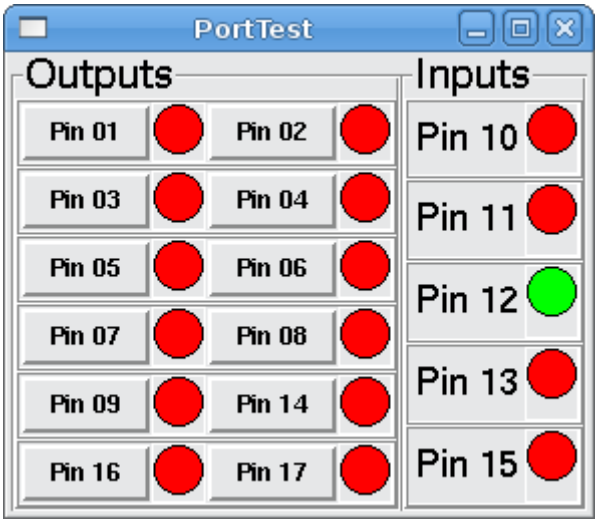
```
loadrt hal_parport cfg="0x378 out"
loadusr -Wn ptest pyvcp -c ptest ptest.xml
loadrt threads name1=porttest period1=1000000
addf parport.0.read porttest
addf parport.0.write porttest
net pin01 ptest.btn01 parport.0.pin-01-out ptest.led-01
net pin02 ptest.btn02 parport.0.pin-02-out ptest.led-02
```

```
net pin10 parport.0.pin-10-in ptest.led-10
net pin11 parport.0.pin-11-in ptest.led-11
start
```

Um die HAL-Datei auszuführen, verwenden wir den folgenden Befehl in einem Terminalfenster.

```
~$ halrun -I -f ptest.hal
```

Die folgende Abbildung zeigt, wie ein komplettes Panel aussehen könnte.



Um die restlichen Pins des Parallelports hinzuzufügen, müssen Sie nur die XML- und HAL-Dateien ändern.

Um die Pins nach der Ausführung des HAL-Skripts anzuzeigen, verwenden Sie den folgenden Befehl an der halcmd-Eingabeaufforderung:

```
halcmd: show pin
Component Pins:
Owner Type  Dir Value  Name
    2 bool  IN  FALSE  parport.0.pin-01-out <== pin01
    2 bool  IN  FALSE  parport.0.pin-02-out <== pin02
    2 bool  IN  FALSE  parport.0.pin-03-out
    2 bool  IN  FALSE  parport.0.pin-04-out
    2 bool  IN  FALSE  parport.0.pin-05-out
    2 bool  IN  FALSE  parport.0.pin-06-out
    2 bool  IN  FALSE  parport.0.pin-07-out
    2 bool  IN  FALSE  parport.0.pin-08-out
    2 bool  IN  FALSE  parport.0.pin-09-out
    2 bool  OUT TRUE   parport.0.pin-10-in ==> pin10
    2 bool  OUT FALSE  parport.0.pin-10-in-not
    2 bool  OUT TRUE   parport.0.pin-11-in ==> pin11
    2 bool  OUT FALSE  parport.0.pin-11-in-not
    2 bool  OUT TRUE   parport.0.pin-12-in
    2 bool  OUT FALSE  parport.0.pin-12-in-not
    2 bool  OUT TRUE   parport.0.pin-13-in
    2 bool  OUT FALSE  parport.0.pin-13-in-not
    2 bool  IN  FALSE  parport.0.pin-14-out
    2 bool  OUT TRUE   parport.0.pin-15-in
```

```
    2 bool  OUT FALSE  parport.0.pin-15-in-not
    2 bool  IN  FALSE  parport.0.pin-16-out
    2 bool  IN  FALSE  parport.0.pin-17-out
    4 bool  OUT FALSE  ptest.btn01 ==> pin01
    4 bool  OUT FALSE  ptest.btn02 ==> pin02
    4 bool  IN  FALSE  ptest.led-01 <== pin01
    4 bool  IN  FALSE  ptest.led-02 <== pin02
    4 bool  IN  TRUE   ptest.led-10 <== pin10
    4 bool  IN  TRUE   ptest.led-11 <== pin11
```

Dies zeigt Ihnen, welche Pins IN und welche Pins OUT sind, sowie alle Verbindungen.

12.2.6. GS2-Drehzahlmesser

Das folgende Beispiel verwendet den Automation Direct GS2 VDF-Treiber und zeigt die U/min (engl. RPM) und andere Informationen in einem PyVCP-Panel an. Dieses Beispiel basiert auf dem GS2-Beispiel im Abschnitt Hardware-Beispiele dieses Handbuchs.

Das Panel

Um das Panel zu erstellen, fügen wir der XML-Datei Folgendes hinzu.

```
<pyvcp>

  <!-- the RPM meter -->
  <hbox>
    <relief>RAISED</relief>
    <bd>3</bd>
    <meter>
      <halpin>"spindle_rpm"</halpin>
      <text>"Spindle"</text>
      <subtext>"RPM"</subtext>
      <size>200</size>
      <min_>0</min_>
      <max_>3000</max_>
      <majorscale>500</majorscale>
      <minorscale>100</minorscale>
      <region1>0,10,"yellow"</region1>
    </meter>
  </hbox>

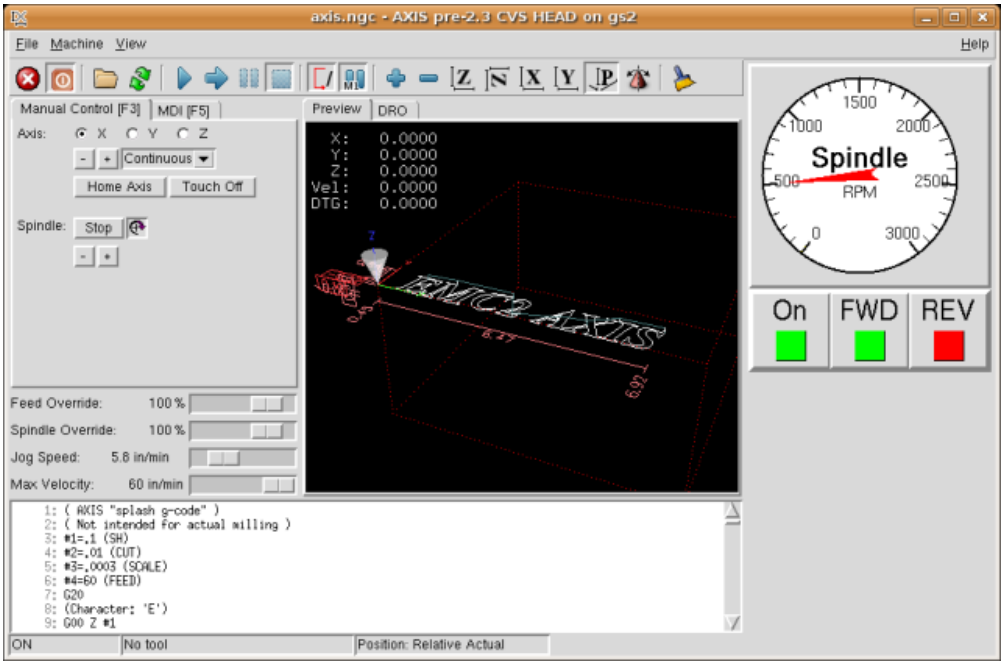
  <!-- the On Led -->
  <hbox>
  <relief>RAISED</relief>
  <bd>3</bd>
  <vbox>
  <relief>RAISED</relief>
  <bd>2</bd>
  <label>
  <text>"On"</text>
  <font>("Helvetica",18)</font>
  </label>
  <width>5</width>
```

```
   <hbox>
  <label width="2"/> <!-- used to center the led -->
  <rectled>
  <halpin>"on-led"</halpin>
  <height>"30"</height>
  <width>"30"</width>
  <on_color>"green"</on_color>
  <off_color>"red"</off_color>
  </rectled>
  </hbox>
  </vbox>

  <!-- the FWD Led -->
  <vbox>
    <relief>RAISED</relief>
    <bd>2</bd>
    <label>
      <text>"FWD"</text>
      <font>("Helvetica",18)</font>
      <width>5</width>
    </label>
   <label width="2"/>
    <rectled>
      <halpin>"fwd-led"</halpin>
      <height>"30"</height>
      <width>"30"</width>
      <on_color>"green"</on_color>
      <off_color>"red"</off_color>
    </rectled>
  </vbox>

  <!-- the REV Led -->
  <vbox>
  <relief>RAISED</relief>
  <bd>2</bd>
    <label>
      <text>"REV"</text>
      <font>("Helvetica",18)</font>
       <width>5</width>
    </label>
    <label width="2"/>
    <rectled>
      <halpin>"rev-led"</halpin>
      <height>"30"</height>
      <width>"30"</width>
      <on_color>"red"</on_color>
      <off_color>"green"</off_color>
    </rectled>
  </vbox>
  </hbox>
</pyvcp>
```

Damit erhalten wir ein PyVCP-Panel, das wie folgt aussieht.



Die Verbindungen

Damit das funktioniert, fügen wir den folgenden Code in die Datei custom_postgui.hal ein.

```
# Anzeige der Drehzahl auf der Grundlage von freq * RPM per Hz
loadrt mult2
addf mult2.0 servo-thread
setp mult2.0.in1 28.75
net cypher_speed mult2.0.in0 <= spindel-vfd.frequency-out
net speed_out pyvcp.spindle_rpm <= mult2.0.out

# run led
net gs2-run => pyvcp.on-led

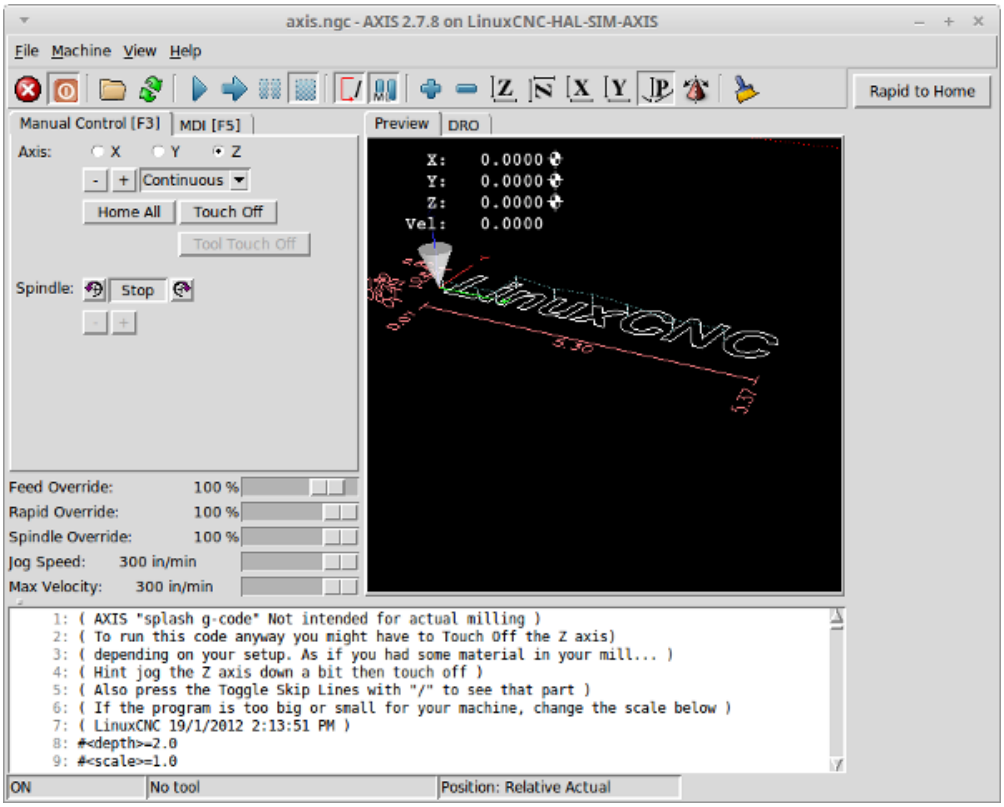
# fwd led
net gs2-fwd => pyvcp.fwd-led

# rev led
net running-rev spindle-vfd.spindle-rev => pyvcp.rev-led
```

Some of the lines might need some explanations. The fwd led line uses the signal created in the custom.hal file whereas the rev led needs to use the spindle-rev bool. You can't link the spindle-fwd bool twice so you use the signal that it was linked to.

12.2.7. Referenzfahrt im Eilgang Button

In diesem Beispiel wird eine Schaltfläche auf dem PyVCP-Seitenpanel erstellt, die bei Betätigung alle Achsen zurück in die Ausgangsposition schickt. Dieses Beispiel setzt voraus, dass Sie kein PyVCP-Panel haben.



Erstellen Sie in Ihrem Konfigurationsverzeichnis die XML-Datei. In diesem Beispiel heißt sie *rth.xml*. Fügen Sie in der Datei "rth.xml" den folgenden Code ein, um die Schaltfläche zu erstellen.

```
<pyvcp>
<!-- rapid to home button example -->
<button>
<halpin>"rth-button"</halpin>
<text>"Rapid to Home"</text>
</button>
</pyvcp>
```

Öffnen Sie Ihre INI-Datei mit einem Texteditor und fügen Sie im Abschnitt [DISPLAY] die folgende Zeile ein. Damit wird das PyVCP-Panel geladen.

```
PYVCP = rth.xml
```

Wenn Sie keinen [HALUI]-Abschnitt in der INI-Datei haben, erstellen Sie ihn und fügen Sie den folgenden MDI-Befehl hinzu.

```
MDI_COMMAND = G53 G0 X0 Y0 Z0
```

NOTE Informationen zu G53 und G0 G-Codes.

Wenn Sie keine Post-GUI-Datei haben, fügen Sie im Abschnitt [HAL] Folgendes hinzu und erstellen Sie eine Datei namens *postgui.hal*.

```
POSTGUI_HALFILE = postgui.hal
```

Fügen Sie in der Datei *postgui.hal* den folgenden Code hinzu, um die PyVCP-Schaltfläche mit dem MDI-Befehl zu verknüpfen.

```
net rth halui.mdi-command-00 <= pyvcp.rth-button
```

NOTE Informationen über den Befehl net

12.3. GladeVCP: Glade Virtuelle Schalttafel

12.3.1. Was ist GladeVCP?

GladeVCP ist eine LinuxCNC-Komponente, welche die Fähigkeit, eine neue Schaltfläche (engl. panel) auf LinuxCNC Benutzeroberflächen hinzuzufügen wie:

- ACHSE
- Touchy
- Gscreen
- GMOCCAPY

Im Gegensatz zu PyVCP ist GladeVCP nicht auf die Anzeige und Einstellung von HAL-Pins beschränkt, da beliebige Aktionen in Python-Code ausgeführt werden können - tatsächlich könnte eine komplette LinuxCNC-Benutzeroberfläche mit GladeVCP und Python erstellt werden.

GladeVCP verwendet den Glade WYSIWYG-Benutzeroberflächen-Editor, der es einfach macht, visuell ansprechende Panels zu erstellen. Es stützt sich auf die PyGObject-Bindungen an den umfangreichen GTK3-Widgetsatz, und tatsächlich können alle diese Widgets in einer GladeVCP-Anwendung verwendet werden - nicht nur die speziellen Widgets für die Interaktion mit HAL und LinuxCNC, die hier dokumentiert sind.

PyVCP im Vergleich zu GladeVCP auf einen Blick

Beide unterstützen die Erstellung von Panels mit "HAL Widgets" - Benutzeroberfläche Elemente wie LED's, Tasten, Schieberegler usw., deren Werte zu einem HAL-Pin, die wiederum Schnittstellen zu den Rest von LinuxCNC verbunden sind.

PyVCP:

- Widget-Set: verwendet TkInter-Widgets.
- Erstellung der Benutzeroberfläche: Zyklus "XML-Datei bearbeiten / Ergebnis ausführen / Aussehen auswerten".
- Keine Unterstützung für die Einbettung benutzerdefinierter Ereignisbehandlung.
- Keine LinuxCNC Interaktion über HAL Pin I/O hinaus unterstützt.

GladeVCP:

- Widgetsatz: stützt sich auf den GTK3-Widgetsatz.
- Erstellung von Benutzeroberflächen: verwendet den Glade WYSIWYG-Editor für Benutzeroberflächen.
- Jede HAL-Pin-Änderung kann für einen Callback und damit einen benutzerdefinierten Python-Ereignishandler genutzt werden.
- Jedes GTK-Signal (Tastendruck, Fenster-, E/A-, Timer-, Netzwerkereignisse) kann mit benutzerdefinierten Handlern in Python verknüpft werden.
- Direkte LinuxCNC-Interaktion: beliebige Befehlsausführung, wie das Auslösen von MDI-Befehlen, um ein G-Code-Unterprogramm aufzurufen, sowie Unterstützung für Statuswechseloperationen durch Action Widgets.
- Mehrere unabhängige GladeVCP-Panels können in verschiedenen Registerkarten ausgeführt werden.
- Trennung von Aussehen und Funktionalität der Benutzeroberfläche: Änderung des Aussehens ohne Eingriff in den Code.

12.3.2. Ein kurzer Rundgang mit dem Beispielpanel

Die GladeVCP-Panel-Fenster können in drei verschiedenen Konfigurationen betrieben werden:

- immer sichtbar, integriert in AXIS auf der rechten Seite, genau wie PyVCP-Panels,
- als Registerkarte in AXIS, Touchy, Gscreen oder GMOCCAPY; in AXIS würde dies eine dritte Registerkarte neben den Registerkarten Vorschau und DRO erzeugen, die explizit angehoben werden müssen,
- als eigenständiges Toplevel-Fenster, das unabhängig vom Hauptfenster ikonifiziert/deikonifiziert werden kann.

Installiertes LinuxCNC

Wenn Sie eine installierte Version von LinuxCNC verwenden, sind die unten gezeigten Beispiele in der configuration picker in der *Sample Configurations* > *apps* > *GladeVCP* Zweig.

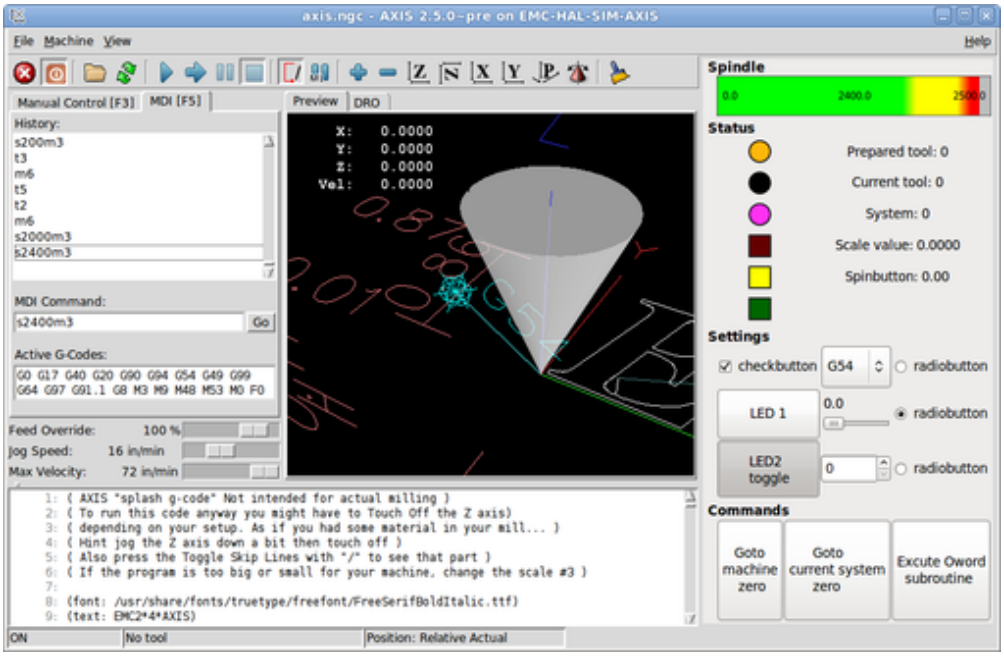
Git-Checkout

Die folgenden Anweisungen gelten nur, wenn Sie einen Git-Checkout verwenden. Öffnen Sie ein Terminal, wechseln Sie in das von git erstellte Verzeichnis und geben Sie dann die angezeigten Befehle ein.

NOTE Damit die folgenden Befehle bei Ihrem Git-Checkout funktionieren, müssen Sie zuerst *make* ausführen, dann *sudo make setuid* und dann *. ./scripts/rip-environment*. Weitere Informationen über einen Git-Checkout finden Sie auf der LinuxCNC-Wiki-Seite.

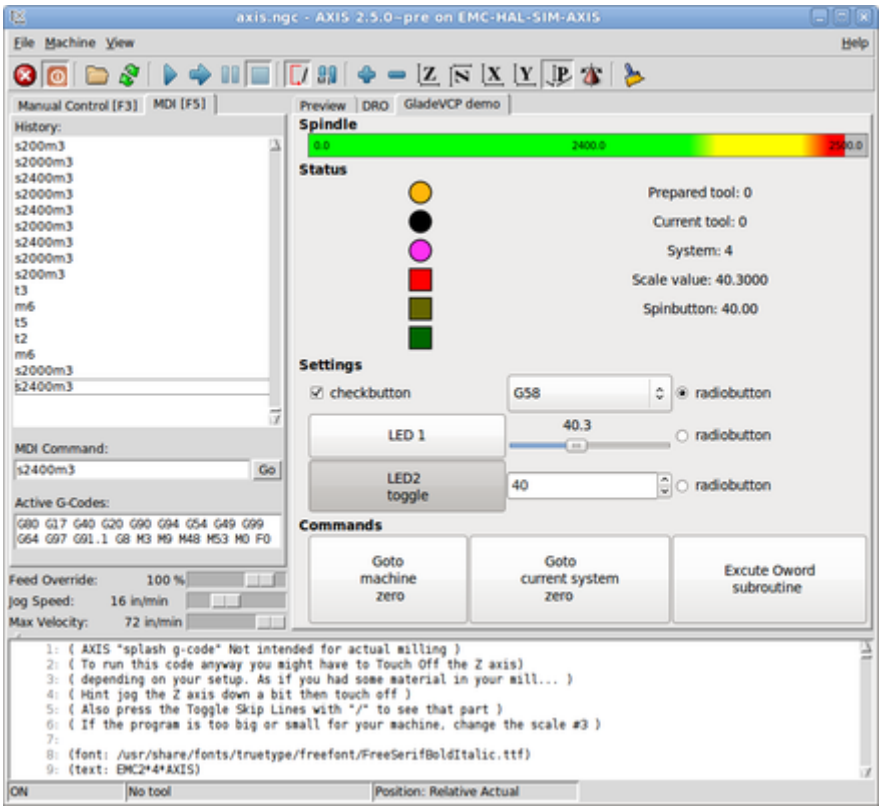
Führen Sie das in AXIS integrierte GladeVCP-Beispielpanel wie PyVCP wie folgt aus:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_panel.ini
```



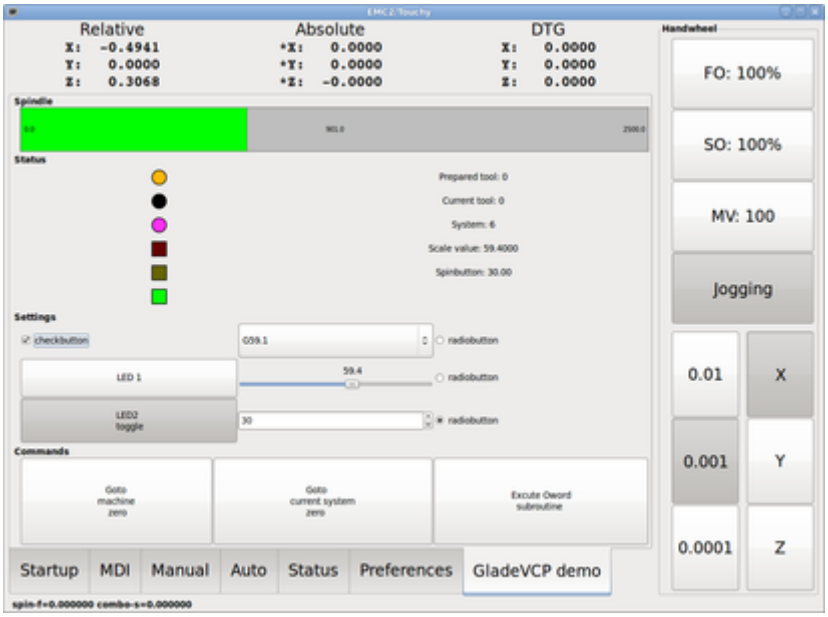
Führen Sie das gleiche Panel aus, aber als Registerkarte innerhalb von AXIS:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_tab.ini
```



Um dieses Panel innerhalb von *Touchy* auszuführen:

```
$ cd configs/sim/touchy/gladevcp
$ linuxcnc gladevcp_touchy.ini
```



Funktional sind diese Setups identisch - sie unterscheiden sich nur in den Anforderungen an die Bildschirmfläche und die Sichtbarkeit. Da es möglich ist, mehrere GladeVCP-Komponenten parallel laufen zu lassen (mit unterschiedlichen HAL-Komponentennamen), sind auch gemischte Setups möglich - zum Beispiel ein Panel auf der rechten Seite und eine oder mehrere Registerkarten für weniger häufig genutzte Teile der Oberfläche.

Erkunden des Beispielpanels

Wenn Sie configs/sim/axis/gladevcp_panel.ini oder configs/sim/axis/gladevcp_tab.ini aufrufen, erkunden Sie *Zeige HAL Konfiguration* (engl. Show HAL Configuration) - Sie finden die `gladevcp`-HAL-Komponente und können ihre Pin-Werte beobachten, während Sie mit den Widgets im Panel interagieren. Die HAL-Konfiguration ist in *configs/axis/gladevcp/manual-example.hal* zu finden.

Das Beispiel-Panel hat zwei Rahmen am unteren Rand. Das Bedienfeld ist so konfiguriert, dass das Zurücksetzen des Notaus (engl. ESTOP) den Rahmen "Einstellungen" aktiviert und das Einschalten der Maschine den Rahmen "Befehle" im unteren Bereich aktiviert. Die HAL-Widgets im Rahmen "Einstellungen" sind mit den LEDs und Beschriftungen im Rahmen "Status" sowie mit der aktuellen und vorbereiteten Werkzeugnummer verknüpft - spielen Sie mit ihnen, um die Wirkung zu sehen. Wenn Sie die Befehle *T<Werkzeugnummer>* und *M6* im MDI-Fenster ausführen, werden die Felder für die aktuelle und die vorbereitete Werkzeugnummer geändert.

Die Buttons im Rahmen "Befehle" (engl. Commands) sind *MDI-Aktions-Widgets* - wenn Sie sie drücken, wird ein MDI-Befehl im Interpreter ausgeführt. Der dritte Button "Oword-Unterprogramm ausführen" (engl. Execute Oword subroutine) ist ein fortgeschrittenes Beispiel - dieser übernimmt mehrere HAL-Pin-Werte aus dem Rahmen *Einstellungen* (engl. Settings) und übergibt sie als Parameter an das Oword-Unterprogramm. Die tatsächlichen Parameter der Routine werden durch *(DEBUG,)* Befehle angezeigt - siehe *../../nc_files/oword.ngc* für den Unterprogrammkörper.

Um zu sehen, wie das Panel in AXIS integriert ist, sehen Sie die Anweisung *[DISPLAY]GLADEVCP* in configs/sim/axis/gladevcp/gladevcp_panel.ini, die Anweisung *[DISPLAY]EMBED** in configs/sim/axis/gladevcp/gladevcp_tab.ini und *[HAL]POSTGUI_HALFILE* sowohl in configs/sim/axis/gladevcp/gladevcp_tab.ini als auch in configs/sim/axis/gladevcp/gladevcp_panel.ini.

Erkunden der Beschreibung der Benutzeroberfläche

Die Benutzeroberfläche wird mit dem Glade UI-Editor erstellt - um sie zu erkunden, müssen Sie Glade installiert haben. Um die Benutzeroberfläche zu bearbeiten, nutzen Sie den Befehl

```
$ glade configs/axis/gladevcp/manual-example.ui
```

Das erforderliche glade-Programm kann auf neueren Systemen den Namen glade-gtk2 tragen.

Das mittlere Fenster zeigt das Aussehen der Benutzeroberfläche. Alle Objekte der Benutzeroberfläche und Unterstützungsobjekte befinden sich im rechten oberen Fenster, in dem Sie ein bestimmtes Widget auswählen können (oder indem Sie es im mittleren Fenster anklicken). Die Eigenschaften des ausgewählten Widgets werden im rechten unteren Fenster angezeigt und können dort geändert werden.

Um zu sehen, wie MDI-Befehle von den MDI-Aktions-Widgets weitergegeben werden, erkunden Sie die Widgets, die unter "Aktionen" im Fenster oben rechts aufgeführt sind, und im Fenster unten rechts unter der Registerkarte "Allgemein" die Eigenschaft "MDI-Befehl".

Erkunden der Python Callback Funktionen

Sehen Sie, wie ein Python-Callback in das Beispiel integriert wird:

- In Glade siehe das `hits` Label Widget (ein einfaches GTK+ Widget).
- Im Widget `button1` auf der Registerkarte "Signale" das Signal "gedrückt" suchen, das mit dem Handler "on_button_press" verknüpft ist.
- In hitcounter.py sehen Sie sich die Methode *on_button_press* an und sehen, wie sie die Eigenschaft label im Objekt *hits* setzt.

Dies ist nur ein kurzer Überblick über das Konzept - der Callback-Mechanismus wird im Abschnitt GladeVCP Programming ausführlicher behandelt.

12.3.3. Erstellen und Integrieren einer Glade-Benutzeroberfläche

Voraussetzung ist: Glade-Installation

Um Glade UI Dateien anzuzeigen oder zu verändern, muss Glade 3.38.2 oder höher installiert sein - es wird nicht benötigt, um ein GladeVCP Panel zu starten. Wenn der Befehl `glade` fehlt, installieren Sie ihn mit dem Befehl:

```
$ sudo apt install glade
```

Überprüfen Sie dann die installierte Version, die gleich oder höher als 3.6.7 sein muss:

```
$ glade --version
```

Glade enthält einen internen Python-Interpreter, und es wird nur Python3 unterstützt. Dies gilt für Debian Bullseye, Ubuntu 21 und Mint 21 oder später. Ältere Versionen werden nicht funktionieren, Sie

werden einen Python-Fehler erhalten.

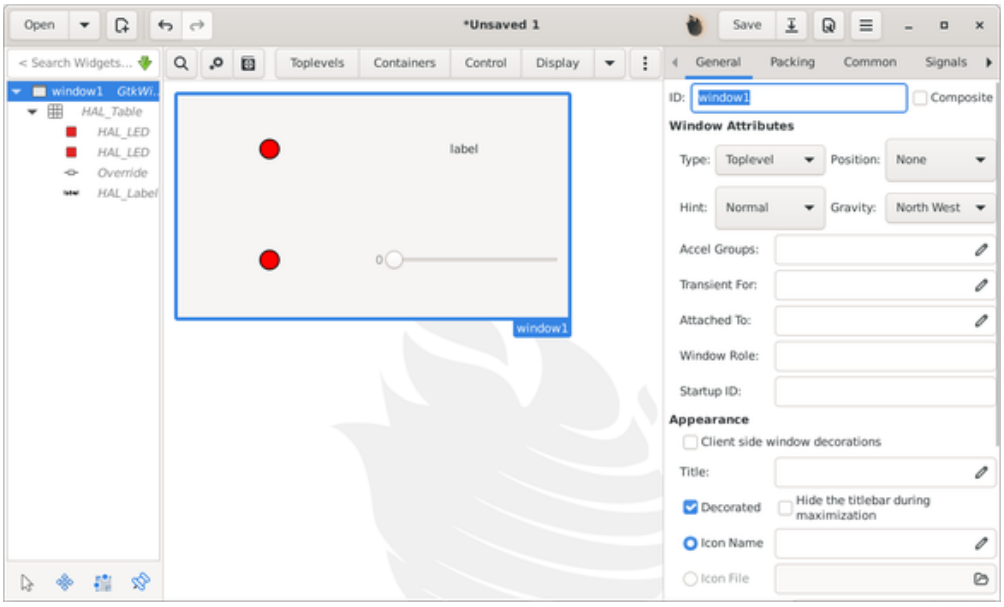
Glade ausführen, um eine neue Benutzeroberfläche zu erstellen

Dieser Abschnitt umreißt nur die ersten LinuxCNC-spezifischen Schritte. Weitere Informationen und eine Anleitung zu Glade finden Sie unter https://glade.gnome.org. Einige Tipps und Tricks zu Glade finden Sie auch auf YouTube.

Ändern Sie entweder eine bestehende UI-Komponente, indem Sie `glade <Datei>.ui` ausführen, oder starten Sie eine neue, indem Sie einfach den Befehl `glade` in der Shell ausführen.

- Wenn LinuxCNC nicht aus einem Paket installiert wurde, muss die LinuxCNC-Shell-Umgebung mit `source <linuxcncdir>/scripts/rip-environment` eingerichtet werden, sonst findet Glade die LinuxCNC-spezifischen Widgets nicht.
- Wenn Sie nach nicht gespeicherten Einstellungen gefragt werden, akzeptieren Sie einfach die Standardeinstellungen und klicken Sie auf „Schließen“.
- Wählen Sie in der Symbolleiste "Toplevels" "GtkWindow" (erster Eintrag) als Fenster der obersten Ebene. Legen Sie *window1* als ID im rechten Fensterbereich unter der Registerkarte *Allgemeines* (engl. *General*) fest. Diese Benennung ist wichtig, weil GladeVCP sich darauf verlässt.
- Über die Schaltfläche mit den drei Punkten können Sie die LinuxCNC-spezifischen Widgets finden.
- Fügen Sie einen Container wie eine HAL_Box oder eine HAL_Table aus *HAL Python* in den Rahmen ein.
- Wählen Sie einige Elemente wie LEDs, Schaltflächen usw. aus und platzieren Sie sie in einem Container.

Dies wird folgendermaßen aussehen:



Glade neigt dazu, eine Menge Meldungen in das Shell-Fenster zu schreiben, die meist ignoriert werden können. Wählen Sie "Datei→Speichern unter", geben Sie der Datei einen Namen wie "myui.ui" und stellen Sie sicher, dass sie als "GtkBuilder"-Datei gespeichert wird (Optionsfeld links unten im Speichern-Dialog). GladeVCP verarbeitet auch das ältere *libglade*-Format korrekt, aber es hat keinen Sinn, es zu

verwenden. Die Konvention für die GtkBuilder-Dateierweiterung ist *.ui*.

Testen eines Panels

Sie sind jetzt bereit, es auszuprobieren (während LinuxCNC, z. B. AXIS läuft) mit:

```
gladevcp myui.ui
```

GladeVCP erstellt eine HAL-Komponente mit dem Namen des Basisnamens der UI-Datei - in diesem Fall *myui* - es sei denn, sie wird mit der Option `-c <Komponentenname>` überschrieben. Wenn AXIS läuft, versuchen Sie einfach *Show HAL configuration* und untersuchen Sie die Pins.

Sie fragen sich vielleicht, warum Widgets, die in einer *HAL_Hbox* oder *HAL_Table* enthalten sind, ausgegraut (inaktiv) erscheinen. HAL-Container haben einen zugehörigen HAL-Pin, der standardmäßig ausgeschaltet ist, so dass alle enthaltenen Widgets inaktiv dargestellt werden. Ein üblicher Anwendungsfall wäre, diese Container-HAL-Pins mit `halui.machine.is-on` oder einem der `halui.mode`-Signale zu verknüpfen, um sicherzustellen, dass einige Widgets nur in einem bestimmten Zustand aktiv erscheinen.

Um nur einen Container zu aktivieren, führen Sie den HAL-Befehl `setp gladevcp.<container-name> 1` aus.

Vorbereiten der HAL-Befehlsdatei

Die vorgeschlagene Methode zur Verknüpfung von HAL-Pins in einem GladeVCP-Panel besteht darin, sie in einer separaten Datei mit der Erweiterung *.hal* zu sammeln. Diese Datei wird über die Option `POSTGUI_HALFILE=` im Abschnitt `HAL` Ihrer INI-Datei übergeben.

CAUTION	Fügen Sie die GladeVCP-HAL-Befehlsdatei nicht in den Abschnitt AXIS `[HAL]HALFILE=` ini ein, da dies nicht den gewünschten Effekt hat - siehe die folgenden Abschnitte.

Einbindung in AXIS, wie PyVCP

Platzieren Sie das GladeVCP-Panel im rechten Seitenbereich, indem Sie in der INI-Datei Folgendes angeben:

```
[DISPLAY]
# add GladeVCP panel where PyVCP used to live:
GLADEVCP= -u ./hitcounter.py ./manual-example.ui

[HAL]
# HAL-Befehle für GladeVCP-Komponenten in einer Registerkarte müssen über POSTGUI_HALFILE
ausgeführt werden
POSTGUI_HALFILE = ./handbuch-beispiel.hal

[RS274NGC]
# gladevcp Demo-spezifische Oword-Subs leben hier
```

```
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Der Standard-HAL-Komponentenname einer GladeVCP-Anwendung, die mit der Option GLADEVCP gestartet wird, lautet: `gladevcp`.

Die von AXIS in der obigen Konfiguration tatsächlich ausgeführte Befehlszeile lautet wie folgt:

```
halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} -u ./hitcounter.py ./manual-
example.ui
```

Sie können hier beliebige `gladevcp`-Optionen hinzufügen, solange sie nicht mit den obigen Kommandozeilenoptionen kollidieren.

Es ist möglich, einen benutzerdefinierten HAL-Komponentennamen zu erstellen, indem Sie die Option `-c` hinzufügen:

```
[DISPLAY]
# fügen Sie das GladeVCP-Panel an der Stelle ein, an der früher PyVCP stand:
GLADEVCP= -c example -u ./hitcounter.py ./manual-example.ui
```

Die Befehlszeile, die von AXIS für die obigen Schritte ausgeführt wird, lautet:

```
halcmd loadusr -Wn example gladevcp -c example -x {XID} -u ./hitcounter.py ./manual-
example.ui
```

NOTE Die Dateispezifikationen wie ./hitcounter.py, ./manual-example.ui usw. bedeuten, dass sich die Dateien im selben Verzeichnis wie die INI-Datei befinden. Möglicherweise müssen Sie sie in Ihr Verzeichnis kopieren (oder einen korrekten absoluten oder relativen Pfad zu der/den Datei(en) angeben).

NOTE Die Option `[RS274NGC]SUBROUTINE_PATH=` ist nur gesetzt, damit das Beispiel-Panel die Oword-Subroutine (oword.ngc) für das MDI Command Widget findet. Sie wird in Ihrem Setup möglicherweise nicht benötigt. Die relative Pfadangabe ../../nc_files/gladevcp_lib ist so konstruiert, dass sie mit Verzeichnissen funktioniert, die von der Konfigurationsauswahl kopiert wurden, und wenn ein Run-in-Place-Setup verwendet wird.

Einbetten als Registerkarte (engl. tab)

Bearbeiten Sie dazu Ihre INI-Datei und fügen Sie die Abschnitte DISPLAY und HAL der INI-Datei wie folgt hinzu:

```
[DISPLAY]
# GladeVCP-Panel als Registerkarte neben Vorschau/DRO hinzufügen:
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} -u
./gladevcp/hitcounter.py ./gladevcp/manual-example.ui
```

```
[HAL]
# HAL-Befehle für GladeVCP-Komponenten in einer Registerkarte müssen über POSTGUI_HALFILE
ausgeführt werden
POSTGUI_HALFILE = ./gladevcp/manual-example.hal

[RS274NGC]
# gladevcp Demo-spezifische Oword-Subs leben hier
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Beachten Sie die `halcmd loadusr`-Art, den Tab-Befehl zu starten - dies stellt sicher, dass *POSTGUI_HALFILE* erst ausgeführt wird, nachdem die HAL-Komponente fertig ist. In seltenen Fällen können Sie hier einen Befehl ausführen, der eine Registerkarte verwendet, aber keine HAL-Komponente zugeordnet ist. Ein solcher Befehl kann ohne *halcmd loadusr* gestartet werden, und dies bedeutet für AXIS, dass es nicht auf eine HAL-Komponente warten muss, da es keine gibt.

Beim Ändern des Komponentennamens im obigen Beispiel ist zu beachten, dass die in `-Wn <Komponente>` und `-c <Komponente>` verwendeten Namen identisch sein müssen.

Probieren Sie es aus, indem Sie AXIS starten - es sollte eine neue Registerkarte mit dem Namen "GladeVCP demo" in der Nähe der Registerkarte "DRO" erscheinen. Wählen Sie diese Registerkarte, sollten Sie das Beispiel-Panel schön innerhalb von AXIS passen zu sehen.

NOTE Stellen Sie sicher, dass die UI-Datei die letzte Option ist, die an GladeVCP in den Anweisungen `GLADEVCP=` und `EMBED_TAB_COMMAND=` übergeben wird.

Integration in Touchy

Um eine GladeVCP-Registerkarte zu "Touchy" hinzuzufügen, bearbeiten Sie Ihre INI-Datei wie folgt:

```
[DISPLAY]
# GladeVCP-Panel als Registerkarte hinzufügen
EMBED_TAB_NAME=GladeVCP-Demo
EMBED_TAB_COMMAND=gladevcp -c gladevcp -x {XID} -u ./hitcounter.py -H ./gladevcp-
touchy.hal ./manual-example.ui

[RS274NGC]
# gladevcp Demo-spezifische Oword-Subs leben hier
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

NOTE Die Dateispezifikationen wie ./hitcounter.py, ./manual-example.ui usw. bedeuten, dass sich die Dateien im selben Verzeichnis wie die INI-Datei befinden. Möglicherweise müssen Sie sie in Ihr Verzeichnis kopieren (oder einen korrekten absoluten oder relativen Pfad zu der/den Datei(en) angeben).

Beachten Sie die folgenden Unterschiede zur Einrichtung der Registerkarte von AXIS:

- Die HAL-Befehlsdatei ist leicht modifiziert, da *Touchy* die *halui*-Komponenten nicht verwendet, so dass seine Signale nicht verfügbar sind und einige Verknüpfungen verwendet wurden.
- Es gibt keine INI-Option *POSTGUI_HALFILE=*, aber die Übergabe der HAL-Befehlsdatei in der Zeile

EMBED_TAB_COMMAND= ist in Ordnung.

- Die Beschwörungsformel "halcmd loaduser -Wn ..." ist nicht erforderlich.

Laden mitgelieferter Panels

There are builtin panels available on the system, you load them in a slightly different way.
You do not add any filename extension to the panel name.
If the panel requires a user file (-u option) it will automatically be loaded.
Builtin panel names can be shown by running the gladevcp command alone in a terminal.

Loading the builtin verser probe panel:

```
gladevcp gtk_verser_probe
```

Embedding is the same, no filename extension, but other options are fine:

```
[DISPLAY]
# GladeVCP-Panel als Registerkarte neben Vorschau/DRO hinzufügen:
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID}
gtk_verser_probe
```

12.3.4. GladeVCP-Befehlszeilenoptionen

These are the GladeVCP command line options:
(See also `man gladevcp`.)
If you enter gladevcp in a terminal this is what you will see:

```
Usage: gladevcp [options] myfile.ui
usage: gladevcp [options] built_in_panel_name

Options:
  -h, --help      show this help message and exit
  -c NAME         Set component name to NAME. Default is basename of UI file
  -d              Enable debug output
  -g GEOMETRY     Set geometry WIDTHxHEIGHT+XOFFSET+YOFFSET. Values are in
                  pixel units, XOFFSET/YOFFSET is referenced from top left of
                  screen use -g WIDTHxHEIGHT for just setting size or -g
                  +XOFFSET+YOFFSET for just position
  -H FILE         execute hal statements from FILE with halcmd after the
                  component is set up and ready
  -i              Enable info output
  -m MAXIMUM      Force panel window to maxumize
  -q              Enable only error debug output
  -r GTK_RC       read custom GTK rc file to set widget style
  -t THEME        Set gtk theme. Default is system theme
  -x XID          Reparent gladevcp into an existing window XID instead of
                  creating a new top level window
  --xid           reparent window into a plug add push the plug xid number to
                  standardout
  -u FILE         Use FILEs as additional user defined modules with handlers
```

```
  -U USEROPT      pass USEROPTs to Python modules
  -v              Enable verbose debug output
  --always_above  Request the window To always be above other windows
  --ini=INI_PATH  ini path

[GladeVCP-][CRITICAL]  Verfügbar für mitgelieferte (built-in) VCP Panels: (gladevcp:205)
['gtk_verser_probe', 'gtk_little_probe']
```

12.3.5. Den GladeVCP-Startvorgang verstehen

Die oben beschriebenen Integrationsschritte sehen ein wenig kompliziert aus, und das sind sie auch. Es ist daher hilfreich, den Startvorgang von LinuxCNC zu verstehen und wie sich dieser auf GladeVCP bezieht.

Der normale Startvorgang von LinuxCNC macht folgendes:

- Die Echtzeitumgebung wird gestartet.
- Alle HAL-Komponenten werden geladen.
- Die HAL-Komponenten sind durch die .hal cmd-Skripte miteinander verbunden.
- `task`, `iocontrol` und schließlich wird die Benutzeroberfläche gestartet.
- Vor der Einführung von GladeVCP wurde davon ausgegangen, dass zu dem Zeitpunkt, zu dem die Benutzeroberfläche gestartet wird, die gesamte HAL geladen, verkabelt und einsatzbereit ist.

Die Einführung von GladeVCP brachte das folgende Problem mit sich:

- GladeVCP-Panels müssen in ein übergeordnetes GUI-Fenster eingebettet werden.
- GladeVCP-Panels müssen in ein übergeordnetes GUI-Fenster eingebettet werden, z.B. AXIS, oder Touchy, Gscreen, oder GMOCCAPY (eingebettetes Fenster oder als eingebettete Registerkarte).
- Dies erfordert, dass die Master-GUI läuft, bevor das GladeVCP-Fenster in die Master-GUI eingehängt werden kann.
- GladeVCP ist jedoch auch eine HAL-Komponente und erzeugt eigene HAL-Pins.
- Als Folge davon müssen alle HAL-Pins, die GladeVCP HAL-Pins als Quelle oder Ziel verwenden, **nach** der Einrichtung der GUI ausgeführt werden.

Dies ist der Zweck der Option `POSTGUI_HALFILE`. Diese INI-Option wird von den GUIs überprüft. Wenn ein GUI diese Option erkennt, führt es die entsprechende HAL-Datei aus, nachdem ein eingebettetes GladeVCP-Panel eingerichtet wurde. Es wird jedoch nicht geprüft, ob ein GladeVCP-Panel tatsächlich verwendet wird, in diesem Fall wird die HAL cmd-Datei einfach normal ausgeführt. Wenn man also GladeVCP *nicht* über `GLADEVCP` oder `EMBED_TAB` usw. startet, sondern später in einem separaten Shell-Fenster oder einem anderen Mechanismus, wird eine HAL-Befehlsdatei in `POSTGUI_HALFILE` zu früh ausgeführt. Unter der Annahme, dass hierin GladeVCP-Pins referenziert werden, wird dies mit einer Fehlermeldung fehlschlagen, die anzeigt, dass die GladeVCP-HAL-Komponente nicht verfügbar ist.

Falls Sie also GladeVCP von einem separaten Shell-Fenster aus starten (d.h. nicht von der GUI eingebettet gestartet):

- Sie können sich nicht darauf verlassen, dass die INI-Option "POSTGUI_HALFILE" dazu führt, dass die HAL-Befehle "zum richtigen Zeitpunkt" ausgeführt werden, also kommentieren Sie sie in der INI-Datei aus.
- Übergeben Sie die HAL-Befehlsdatei, die auf GladeVCP-Pins verweist, explizit an GladeVCP mit der Option `-H <halcmd file>` (siehe vorheriger Abschnitt).

12.3.6. HAL Widget-Referenz

GladeVCP enthält eine Sammlung von Gtk-Widgets mit angehängten HAL-Pins, genannt HAL-Widgets, die zur Steuerung, Anzeige oder anderweitigen Interaktion mit der LinuxCNC-HAL-Schicht gedacht sind. Sie sind für die Verwendung mit dem Glade-Benutzerschnittstellen-Editor vorgesehen. Bei ordnungsgemäßer Installation sollten die HAL Widgets in der Widget-Gruppe "HAL Python" von Glade auftauchen. Viele HAL-spezifische Felder im Glade-Abschnitt "Allgemein" haben einen zugehörigen Tooltip, wenn man mit der Maus darüber fährt.

HAL signals come in two variants, booleans and numbers. BOOLs are off/on signals. Numbers can be "REAL", "SINT" or "UINT". For more information on HAL data types see the HAL manual. The GladeVCP widgets can either display the value of the signal with an indicator widget, or modify the signal value with a control widget. Thus there are four classes of GladeVCP widgets that you can connect to a HAL signal. Another class of helper widgets allow you to organize and label your panel.

- Widgets for indicating "BOOL" signals: HAL_LED
- Widgets for controlling "BOOL" signals: HAL_Button HAL_RadioButton HAL_CheckButton
- Widgets zur Anzeige von "Zahlen"-Signalen: HAL_Label, HAL_ProgressBar, HAL_HBar und HAL_VBar, HAL_Meter
- Widgets zur Steuerung von "Zahlen"-Signalen: HAL_SpinButton, HAL_HScale and HAL_VScale, Jog Wheel, Geschwindigkeitsregelung
- Empfindliche Kontroll-Widgets: State_Sensitive_Table HAL_Table and HAL_HBox
- Vorschau des Werkzeugpfads: HAL_Gremlin
- Widgets zur Anzeige der Achsenpositionen: DRO Widget, Combi DRO Widget
- Widgets für die Dateiverarbeitung: IconView Datei Auswahl
- Widgets zur Anzeige/Bearbeitung aller Achsen-Offsets: Offset-Seite
- Widgets zur Anzeige/Bearbeitung aller Werkzeugversätze: Werkzeug-Versatz-Editor
- Widget zur Anzeige und Bearbeitung von G-Code: HAL_Sourceview
- Widget für MDI-Eingabe und Verlaufsanzeige: MDI History

Benennung von Widgets und HAL-Pins

Die meisten HAL-Widgets haben einen einzigen zugehörigen HAL-Pin mit demselben HAL-Namen wie das Widget (glade: General→Name).

Ausnahmen von dieser Regel sind derzeit:

- *HAL_Spinbutton* and *HAL_ComboBox*, which have two pins: a `<widgetname>-f` (REAL) and a `<widgetname>-s` (SINT) pin
- HAL_ProgressBar', die einen `<widgetname>-value`-Eingangspin und einen `<widgetname>-scale` -Eingangspin hat.

Python-Attribute und Methoden von HAL Widgets

HAL-Widgets sind Instanzen von GtKWidgets und erben daher die Methoden, Eigenschaften und Signale der entsprechenden GtkWidget-Klasse. Um z.B. herauszufinden, welche GtkWidget-bezogenen Methoden, Eigenschaften und Signale ein *HAL_Button* hat, sehen Sie in der Beschreibung von GtkButton in der PyGObject API Reference nach.

Ein einfacher Weg, um die Vererbungsbeziehung eines bestimmten HAL-Widgets herauszufinden, ist folgender: Starten Sie Glade, platzieren Sie das Widget in einem Fenster und wählen Sie es aus; wählen Sie dann die Registerkarte *Signale* im Fenster *Eigenschaften*. Wenn Sie zum Beispiel ein *HAL_LED*-Widget auswählen, wird dies zeigen, dass ein *HAL_LED* von einem *GtkWidget* abgeleitet ist, welches wiederum von einem *GtkObject* und schließlich einem *GObject* abgeleitet ist.

Die vollständige Klassenhierarchie kann durch Aufrufen des GtkInspectors in der Glade-GUI eingesehen werden, indem man ein Widget auswählt und dann Control-Shift-I drückt. Wenn sich der Inspector nicht öffnet, kann er von einem Terminal aus durch Eingabe von:

```
gsettings set org.gtk.Settings.Debug enable-inspector-keybinding true
```

Der Inspektor ist auch praktisch, um CSS-Stiländerungen "on the fly" zu testen und alle für ein Widget verfügbaren Eigenschaften und Signale zu ermitteln.

HAL-Widgets haben auch einige HAL-spezifische Python-Attribute:

hal_pin

Das zugrundeliegende HAL-Pin-Python-Objekt, falls das Widget einen einzigen Pin-Typ hat

hal_pin_s, hal_pin_f

The SINT and REAL pins of the *HAL_Spinbutton* and *HAL_ComboBox* widgets - note these widgets do not have a *hal_pin* attribute!

hal_pin_scale

The REAL input pin of *HAL_ProgressBar* widget representing the maximum absolute value of input.

Es gibt mehrere HAL-spezifische Methoden von HAL Widgets, aber die einzige relevante Methode ist:

<halpin>.get()

Abrufen des Wertes des aktuellen HAL-Pins, wobei <*halpin*> der oben aufgeführte zutreffende HAL-Pinname ist.

Einstellung von Pin- und Widget-Werten

Als allgemeine Regel gilt: Wenn Sie den Wert eines HAL-Ausgabe-Widgets von Python-Code aus setzen müssen, tun Sie dies, indem Sie den zugrundeliegenden Gtk-*Setter* (z.B. `set_active()`, `set_value()`) aufrufen . Versuchen Sie nicht, den Wert des zugehörigen Pins direkt durch `halcomp[pinname] = value` zu setzen, da das Widget die Änderung nicht zur Kenntnis nimmt!

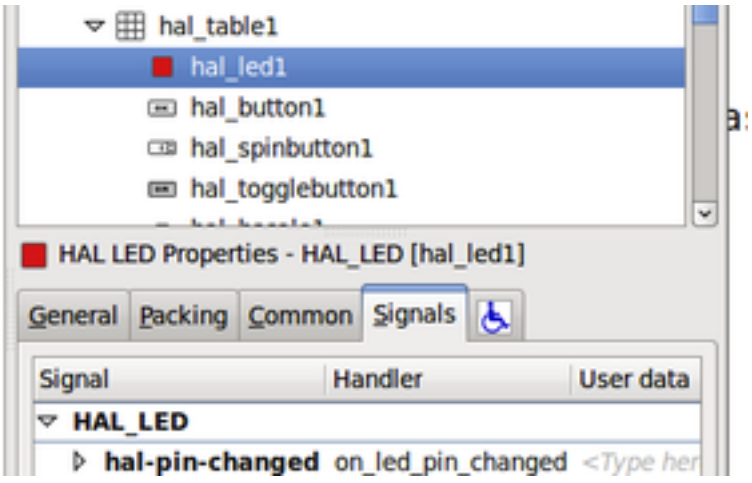
Es mag verlockend sein, HAL-Widget-Eingangs-Pins programmatisch zu setzen. Beachten Sie, dass dies den Zweck eines Eingangspins von vornherein zunichte macht - er sollte mit anderen HAL-Komponenten verknüpft sein und auf von ihnen erzeugte Signale reagieren. Da es derzeit keinen Schreibschutz für das Schreiben auf Input-Pins in HAL Python gibt, macht dies keinen Sinn. Zum Testen können Sie aber `setp _pinname_ _value_` in der zugehörigen HAL-Datei verwenden.

Es ist völlig in Ordnung, den Wert eines Ausgangs-HAL-Pins mit `halcomp[pinname] = value` zu setzen, vorausgesetzt, dieser HAL-Pin ist nicht mit einem Widget verbunden, d.h. er wurde mit der Methode `hal_glib.GPin(halcomp.newpin(<name>,<type>,<direction>))` erstellt (siehe GladeVCP Programming für ein Beispiel).

Das hal-pin-changed-Signal

Ereignisgesteuerte Programmierung bedeutet, dass die Benutzeroberfläche Ihrem Code mitteilt, wenn "etwas passiert" - durch einen Rückruf, z. B. wenn eine Taste gedrückt wurde. Die Ausgabe-HAL-Widgets (die den Wert eines HAL-Pins anzeigen) wie LED, Balken, VStabi, Zähler usw. unterstützen das Signal `hal-pin-changed`, das einen Callback in Ihrem Python-Code auslösen kann, wenn - nun ja, ein HAL-Pin seinen Wert ändert. Das bedeutet, dass es keine Notwendigkeit mehr gibt, HAL-Pin-Änderungen in Ihrem Code permanent abzufragen, die Widgets erledigen das im Hintergrund und informieren Sie darüber.

Hier ist ein Beispiel, wie man ein `hal-pin-changed` Signal für eine HAL_LED im Glade UI Editor setzt:



Das Beispiel in `configs/apps/gladevcp/complex` zeigt, wie dies in Python gehandhabt wird.

Buttons

This group of widgets are derived from various Gtk buttons and consists of HAL_Button, HAL_ToggleButton, HAL_RadioButton and CheckButton widgets. All of them have a single output BOOL

pin named identical to the widget. Buttons have no additional properties compared to their base Gtk classes.

- HAL_Button: Sofortige Aktion, behält den Zustand nicht bei. Wichtiges Signal: `pressed` (engl. für gedrückt)
- HAL_ToggleButton, HAL_CheckButton: behält den Ein/Aus-Zustand bei. Wichtiges Signal: `toggled` (engl. für umgeschaltet)
- HAL_RadioButton: eine "One-of-many"-Gruppe. Wichtiges Signal: `toggled`(umgeschaltet) (pro Button).
- Wichtige gemeinsame Methoden: `set_active()`, `get_active()`
- Wichtige Eigenschaften: `label`, `image`

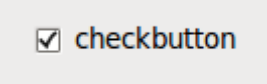


Figure 251. Checkbutton

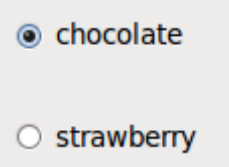


Figure 252. Radiobuttons

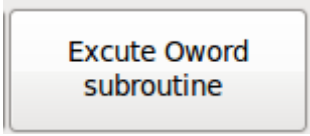


Figure 253. Toggle-Button

TIP

Definieren von Optionsschaltflächengruppen in Glade:

- Entscheiden Sie sich für einen aktiven Standardbutton.
- Wählen Sie unter *Allgemein→Gruppe* der anderen Schaltfläche den Namen der standardmäßig aktiven Schaltfläche im Dialogfeld *Wählen Sie eine Optionsschaltfläche in diesem Projekt*.

Siehe `configs/apps/gladevcp/by-widget/` für eine GladeVCP-Anwendung und UI-Datei für die Arbeit mit Optionsfeldern.

Skalen (engl. scales)

HAL_HScale und HAL_VScale sind von GtkHScale bzw. GtkVScale abgeleitet.

<widgetname>

out REAL pin

<widgetname>-s

out SINT pin

Um eine Skala in Glade nützlich zu machen, fügen Sie eine "Anpassung" hinzu (Allgemein → Anpassung → Neue oder bestehende Anpassung) und bearbeiten das Anpassungsobjekt. Es definiert die Standard-/Min-/Max-/Inkrementwerte. Setzen Sie außerdem die Anpassung *Seitengröße* und *Seiteninkrement* auf Null, um Warnungen zu vermeiden.

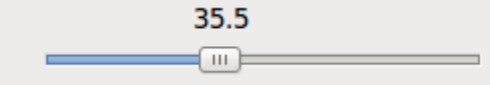


Figure 254. Beispiel HAL_HScale

SpinButton

HAL SpinButton ist von GtkSpinButton abgeleitet und besitzt zwei Pins:

<widgetname>-f

out REAL pin

<widgetname>-s

out SINT pin

Um nützlich zu sein, benötigen SpinButtons einen Einstellwert wie Skalen, siehe oben.

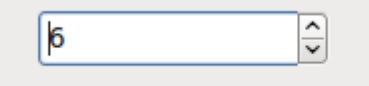


Figure 255. Beispiel SpinButton

Hal_Dial

Das hal_dial Widget simuliert ein Jogwheel oder ein Einstellrad.
Es kann mit der Maus bedient werden. Sie können einfach das Mausrad benutzen, während sich der Mauszeiger über dem Hal_Dial-Widget befindet, oder Sie halten die linke Maustaste gedrückt und bewegen den Mauszeiger in kreisförmiger Richtung, um die Zählungen zu erhöhen oder zu verringern.
Mit einem Doppelklick auf die linke oder rechte Taste kann der Skalierungsfaktor erhöht oder verringert werden.

- Gegen den Uhrzeigersinn = kleinere Zählwerte
- Im Uhrzeigersinn = erhöht Zähler
- Rad hoch = erhöht Zähler
- Rad nach unten = erniedrigt Zähler
- Links Doppelklick = x10 Skalierung
- Rechter Doppelklick = /10 Skalierung

Pins

`hal_dial` exportiert seinen Zählwert als HAL-Pins:

<widgetname>

out SINT pin

<widgetname>-scaled

out REAL pin

<widgetname>-delta-scaled

out REAL pin

Eigenschaften

`hal_dial` hat die folgenden Eigenschaften:

cpr

Legt die Anzahl der Zählungen pro Umdrehung fest, zulässige Werte liegen im Bereich von 25 bis 360 +
Voreinstellung = 100

show_counts

Setzen Sie diesen Wert auf False, wenn Sie die Anzeige der Zählerstände in der Mitte des Widgets ausblenden möchten. +
Standard = True

label

Legen Sie den Inhalt der Beschriftung fest, die über dem Zählwert angezeigt werden kann.
Ist die angegebene Beschriftung länger als 15 Zeichen, wird sie auf 15 Zeichen gekürzt.
Standard = leer

center_color

Damit kann man die Farbe des Rades ändern. Sie verwendet einen GDK-Farbstring. +
Standard = #bdefbdefbdef (grau)

count_type_shown

Es sind drei Zählungen verfügbar 0) Rohe CPR-Zählungen 1) Skalierte Zählungen 2) Delta-skalierte Zählungen. +
Standard = 1

- count is based on the CPR selected - it will count positive and negative. It is available as a SINT pin.
- Scaled-count is CPR count times the scale - it can be positive and negative.
  If you change the scale the output will immediately reflect the change. It is available as a REAL pin.
- Delta-scaled-count is cpr count CHANGE, times scale.
  If you change the scale, only the counts after that change will be scaled and then added to the

current value.
It is available as a REAL pin.

scale_adjustable

Setzen Sie diesen Wert auf False, wenn Sie Änderungen der Skalierung durch Doppelklick auf das Widget nicht zulassen wollen. +
Wenn dieser Wert auf False gesetzt ist, wird der Skalierungsfaktor im Widget nicht angezeigt. +
Standard = True

scale

Legen Sie diesen Wert fest, um die Zählungen zu skalieren. +
Standard = 1.0

Direkte Programmsteuerung

Es gibt Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie goobject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property("cpr",int(value))
[widget name].set_property("show_counts, True)
[widget name].set_property("center_color",gtk.gdk.Color('#bdefbdefbdef'))
[widget name].set_property('label', 'Test Dial 12345')
[widget name].set_property('scale_adjustable', True)
[widget name].set_property('scale', 10.5)
[widget name].set_property('count_type_shown', 0)
```

Es gibt Python-Methoden:

- `[widget name].get_value()`
  Will return the counts value as an integer
- [Widgetname].get_scaled_value() `
  Gibt den Zählwert als Gleitkommazahl zurück
- *[Widget-Name].get_delta_scaled_value()* +
  Gibt den Counts-Wert als Gleitkommawert zurück
- `[Widgetname].set_label("string")` +
  Setzt den Inhalt des Labels mit "string"

Es werden zwei GObject-Signale ausgegeben:

- `count_changed` +
  Wird ausgesendet, wenn sich die Anzahl des Widgets ändert, z.B. wenn es mit dem Rad gescrollt wird.
- `scale_changed`
  Wird ausgegeben, wenn sich der Maßstab des Widgets ändert, z. B. durch Doppelklick.

Schließen Sie diese wie folgt an:

```
[widget name].connect('count_changed', [count function name])
[widget name].connect('scale_changed', [scale function name])
```

Die Callback-Funktionen würden dieses Muster verwenden:

```
def [Name der Zählfunktion](widget, count,scale,delta_scale):
```

Dies gibt zurück: das Widget, die aktuelle Anzahl, Skalierung und Delta-Skalierung dieses Widgets.

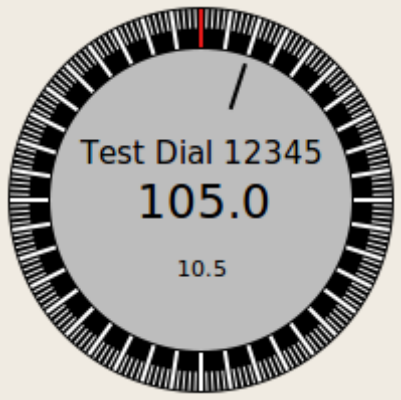


Figure 256. Beispiel Hal_Dial

Jog-Handrad (engl. jog wheel)

Das Widget `jogwheel` simuliert ein echtes Jogwheel. Es kann mit der Maus bedient werden. Sie können einfach das Mausrad benutzen, während sich der Mauszeiger über dem JogWheel-Widget befindet, oder Sie drücken die linke Maustaste und bewegen den Mauszeiger in kreisförmiger Richtung, um die Zählungen zu erhöhen oder zu verringern.

- Gegen den Uhrzeigersinn = kleinere Zählwerte
- Im Uhrzeigersinn = erhöht Zähler
- Rad hoch = erhöht Zähler
- Rad nach unten = erniedrigt Zähler

Da das Bewegen der Maus per Drag & Drop schneller sein kann, als das Widget sich selbst zu aktualisieren vermag, kann, dass Sie Zähler verlieren, wenn Sie zu schnell drehen. Es wird empfohlen, das Mausrad zu verwenden, und nur für sehr grobe Bewegungen die Drag-and-Drop-Methode.

Pins

`jogwheel` exportiert seinen Zählwert als HAL-Pin:

<widgetname>-s

out SINT pin

Eigenschaften

`jogwheel` hat folgende Eigenschaften:

Größe

Legt die Größe des Widgets in Pixeln fest; die zulässigen Werte liegen im Bereich von 100 bis 500 Standard = 200

cpr

Legt die Anzahl der Zählungen pro Umdrehung fest, zulässige Werte liegen im Bereich von 25 bis 100 Standard = 40

show_counts

Setzen Sie diesen Wert auf False, wenn Sie die Anzeige der Zählerstände in der Mitte des Widgets ausblenden möchten.

label

Legen Sie den Inhalt der Beschriftung fest, die über dem Zählwert angezeigt werden kann. Der Zweck ist, dem Benutzer eine Vorstellung über die Verwendung dieses Jogwheels zu geben. Wenn die Beschriftung länger als 12 Zeichen ist, wird sie auf 12 Zeichen gekürzt.

Direkte Programmsteuerung

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie GObject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property("size",int(value))
[widget name].set_property("cpr",int(value))
[widget name].set_property("show_counts, True)
```

Es gibt zwei Python-Methoden:

- `[widget name].get_value()`
  Gibt den Zählwert als Ganzzahl zurück
- `[Widgetname].set_label("string")` +
  Setzt den Inhalt des Labels mit "string"

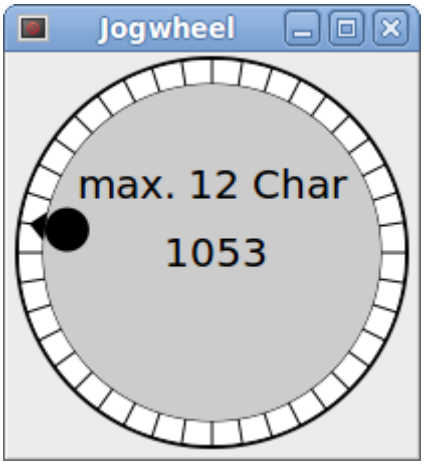


Figure 257. Beispiel JogWheel

Geschwindigkeitsregelung

`speedcontrol` ist ein Widget, das speziell für die Steuerung einer Einstellung über einen Touchscreen entwickelt wurde. Es ist ein Ersatz für das normale Skalen-Widget, das auf einem Touchscreen nur schwer zu verschieben ist.

Der Wert wird mit zwei Tasten zum Erhöhen oder Verringern des Wertes gesteuert. Die Schrittweite ändert sich, solange eine Schaltfläche gedrückt wird. Der Wert jedes Inkrements sowie die Zeit zwischen zwei Änderungen können über die Widgeteigenschaften eingestellt werden.

Pins

speedcontrol bietet einige HAL-Pins:

<widgetname>-value

out REAL pin
The shown value of the widget.

<widgetname>-scaled-value

out REAL pin
The shown value divided by the scale value, this is very useful, if the velocity is shown in units / min, but LinuxCNC expects it to be in units / second.

<widgetname>-scale

in REAL pin
The scale to apply.
Default is 60.

<widgetname>-increase

in BOOL pin
As long as the pin is true, the value will increase.
Very handy with connected momentary switch.

<widgetname>-decrease

in BOOL pin
As long as the pin is true, the value will decrease.
Very handy with connected momentary switch.

Eigenschaften

speedcontrol hat die folgenden Eigenschaften:

height (engl. für Höhe)

Integer +
Die Höhe des Widgets in Pixel. +
Erlaubte Werte sind 24 bis 96. +
Voreinstellung ist 36.

Wert

Gleitkommazahl +
Der einzustellende Startwert. +
Erlaubte Werte liegen im Bereich von 0,001 bis 99999,0. +
Standardwert ist 10,0.

min

Gleitkommazahl +
Der zulässige Mindestwert. +
Erlaubte Werte sind 0,0 bis 99999,0. +
Der Standardwert ist 0,0. +
Wenn Sie diesen Wert ändern, wird die Schrittweite auf den Standardwert zurückgesetzt, so dass es notwendig sein kann, anschließend eine neue Schrittweite festzulegen.

max

Gleitkommazahl
Der maximal zulässige Wert.
Erlaubte Werte sind 0,001 bis 99999,0.
Der Standardwert ist 100,0.
Wenn Sie diesen Wert ändern, wird die Schrittweite auf den Standardwert zurückgesetzt, so dass es notwendig sein kann, anschließend eine neue Schrittweite festzulegen.

increment (engl. für Zunahme)

Gleitkomma +
Legt die angewandte Schrittweite pro Mausklick fest. +
Erlaubte Werte sind 0,001 bis 99999,0 und -1. +
Standardwert ist -1, was zu 100 Schritten von min bis max führt.

inc_speed

Ganzzahlig +
Legt die Zeitverzögerung für die Erhöhung der Geschwindigkeit fest, bei der die Tasten gedrückt gehalten werden. +

Erlaubte Werte sind 20 bis 300. +
Voreinstellung ist 100.

unit (engl. für Einheit)

Zeichenfolge
Legt die Einheit fest, die in der Leiste hinter dem Wert angezeigt wird.
Jede Zeichenkette ist erlaubt.
Standard ist "".

color (engl. für Farbe)

Farbe +
Legt die Farbe des Balkens fest. +
Jede Hex-Farbe ist erlaubt. +
Standard ist "#FF8116".

template (engl. für Vorlage)

Zeichenfolge +
Textvorlage zur Anzeige des Wertes. Es wird die Python-Formatierung verwendet. +
Jedes zulässige Format. +
Standard ist "%.1f".

do_hide_button

Boolescher Wert +
Ob die Schaltfläche zum Erhöhen und Verringern angezeigt oder ausgeblendet werden soll. +
True oder False. +
Voreinstellung = False.

Direkte Programmsteuerung

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie GObject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property("do_hide_button",bool(value))
[widget name].set_property("color","#FF00FF")
[widget name].set_property("unit", "mm/min")
usw.
```

Es gibt auch Python-Methoden zur Änderung des Widgets:

```
[widget name].set_adjustment(gtk-adjustment)
```

Sie können dem Regler eine bestehende Einstellung zuweisen, so dass es einfach ist, bestehende Schieberegler ohne viele Codeänderungen zu ersetzen. Beachten Sie, dass Sie nach dem Ändern der Einstellung möglicherweise eine neue Schrittweite einstellen müssen, da sie auf die Standardeinstellung zurückgesetzt wird (100 Schritte von MIN bis MAX):

- `[Widgetname].get_value()` +

Gibt den Zählwert als Float zurück

- `[Widgetname].set_value(float(Wert))` +
  Setzt das Widget auf den befohlenen Wert
- `[Widgetname].set_digits(int(Wert))` +
  Setzt die Ziffern des zu verwendenden Wertes
- `[Widgetname].hide_button(bool(Wert))` +
  Die Schaltfläche ausblenden oder anzeigen

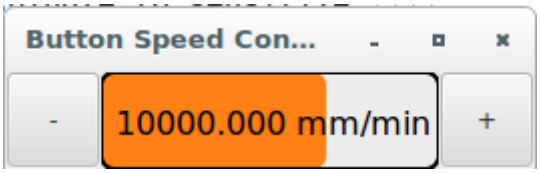


Figure 258. Beispiel Speedcontrol

Label

`hal_label` ist ein einfaches Widget, das auf GtkLabel basiert und einen HAL-Pin-Wert in einem benutzerdefinierten Format darstellt.

label_pin_type

The pin's HAL type (0:SINT, 1:REAL, 2:UINT), see also the tooltip on *General→HAL pin type* (note this is different from PyVCP which has three label widgets, one for each type).

text_template

Determines the text displayed - a Python format string to convert the pin value to text. Defaults to `%s` (values are converted by the str() function) but may contain any legit as an argument to Pythons format() method.
Example: `Distance: %.03f` will display the text and the pin value with 3 fractional digits padded with zeros for a REAL pin.

Containers (engl. für Behälter)

- HAL_HideTable
- HAL_Table
- State_Sensitive_Table
- HAL_HBox (veraltet)

Diese Container sollen dazu dienen, ihre Kinder zu insensibilisieren (auszugrauen) oder zu verstecken. +
Nicht sensibilisierte Kinder reagieren nicht auf Eingaben.

HAL_HideTable

Has one HAL BOOL input pin which controls if its child widgets are hidden or not.

Pin:

`<Panel_basename>. <widgetname>`

in BOOL pin
If the pin is low then child widgets are visible which is the default state.

HAL_Table und HAL_Hbox

Have one HAL BOOL input pin which controls if their child widgets are sensitive or not.

Pin:

`<Panel_basename>. <widgetname>`

in BOOL pin
If the pin is low then child widgets are inactive which is the default state.

State_Sensitive_Table

Reagiert auf den Zustand der LinuxCNC's Interpreter.
Optional auswählbar, um auf *must-be-all-homed*, *must-be-on* und *must-idle* zu reagieren.
Sie können sie kombinieren. Bei Notaus (engl. E-stop) ist es immer unempfindlich.
(Hat keinen Pin).

WARNING **HAL_Hbox ist veraltet - verwenden Sie HAL_Table**.
Wenn aktuelle Panels es verwenden, wird es nicht scheitern. Sie werden es nur nicht mehr im GLADE-Editor finden. +
Zukünftige Versionen von GladeVCP könnten dieses Widget komplett entfernen und dann müssen Sie das Panel aktualisieren.

TIP Wenn Sie feststellen, dass ein Teil Ihrer GladeVCP-Anwendung "ausgegraut" (unempfindlich) ist, prüfen Sie, ob ein HAL_Table-Pin nicht gesetzt oder nicht angeschlossen ist.

LED

The `hal_led` simulates a real indicator LED.
It has a single input BOOL pin which controls its state: ON or OFF.

Eigenschaften

LEDs haben verschiedene Eigenschaften, die ihr Aussehen und ihre Wirkung bestimmen:

on_color

String, der die ON-Farbe der LED definiert. +
Kann jeder gültige gdk.Color-Name sein. +
Funktioniert nicht unter Ubuntu 8.04.

off_color

String, der die OFF-Farbe der LED definiert. +
Kann jeder gültige gdk.Color-Name oder der spezielle Wert `dark` sein. `dark` bedeutet, dass die OFF-

Farbe auf den Wert 0,4 der ON-Farbe gesetzt wird.
Funktioniert nicht unter Ubuntu 8.04.

pick_color_on, pick_color_off

Farben für den EIN- und AUS-Zustand.
Diese können als "#RRRRGGGGBBBB"-Strings dargestellt werden und sind optionale Eigenschaften, die Vorrang vor "on_color" und "off_color" haben.

led_size

LED-Radius (für Quadrat - halbe LED-Seite)

led_shape

LED-Form. +
Gültige Werte sind 0 für runde, 1 für ovale und 2 für quadratische Formen.

led_blink_rate

Wenn gesetzt und die LED eingeschaltet ist, blinkt sie.
Die Blinkperiode ist gleich der "led_blink_rate", die in Millisekunden angegeben wird.

create_hal_pin

Aktivierung/Deaktivierung der Erstellung eines HAL-Pins zur Steuerung der LED. +
Wenn kein HAL-Pin erstellt wurde, kann die LED mit einer Python-Funktion gesteuert werden.

Signale

Als Input-Widget unterstützt die LED auch das Signal `hal-pin-changed`. Wenn Sie in Ihrem Code eine Benachrichtigung erhalten möchten, wenn der HAL-Pin der LED geändert wurde, dann verbinden Sie dieses Signal mit einem Handler, z.B. `on_led_pin_changed`, und stellen Sie den Handler wie folgt bereit:

```
def on_led_pin_changed(self,hal_led,data=None):
    print("on_led_pin_changed() - HAL Pin Wert:",hal_led.hal_pin.get())
```

Dieser wird bei jeder Flanke des Signals und auch beim Programmstart aufgerufen, um den aktuellen Wert zu melden.

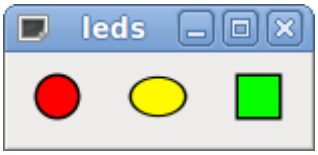


Figure 259. Beispiel LEDs

ProgressBar (engl. für Fortschrittsbalken)

NOTE	Dieses Widget wird möglicherweise entfernt. Verwenden Sie stattdessen die Widgets HAL_HBar und HAL_VBar.

Pins

The `HAL_ProgressBar` is derived from gtk.ProgressBar and has two REAL HAL input pins:

<widgetname>

den aktuell anzuzeigenden Wert

<widgetname>-scale

der maximale absolute Wert der Eingabe

Eigenschaften

`HAL_ProgressBar` hat die folgenden Eigenschaften:

scale

Werteskala. +
Legt den maximalen absoluten Wert der Eingabe fest. Entspricht dem Setzen des <widgetname>.scale-Pins.
Ein Float, Bereich von -2^24 bis +2^24.

green_limit

Untere Grenze der grünen Zone

yellow_limit

Untere Grenze der gelben Zone

red_limit

Untergrenze der roten Zone

text_template

Textvorlage zur Anzeige des aktuellen Wertes des `__<widgetname>__` Pin.
Python-Formatierung kann für dict `{"value":value}` verwendet werden.



Figure 260. Beispiel HAL_ProgressBar

ComboBox

Die `HAL_ComboBox` ist von der gtk.ComboBox abgeleitet. Sie ermöglicht die Auswahl eines Wertes aus einer Dropdown-Liste.

Pins

Die `HAL_ComboBox` exportiert zwei HAL-Pins:

<widgetname>-f

Current value, type REAL

<widgetname>-s

Current value, type SINT

Eigenschaften

`HAL_ComboBox` hat die folgende Eigenschaft, die in Glade gesetzt werden kann:

column

The column index.
Type SINT.
Valid range from -1..100.
Defaults value -1.

Im Standardmodus setzt dieses Widget die Pins auf den Index des gewählten Listeneintrags. Wenn Ihr Widget also drei Etiketten hat, kann es nur die Werte 0, 1 und 2 annehmen.

In column mode (column > -1), the value reported is chosen from the ListStore array as defined in Glade. So typically your widget definition would have two columns in the ListStore, one with text displayed in the dropdown, and an integer or floating point value to use for that choice.

Es gibt ein Beispiel in `configs/apps/by-widget/combobox. {py,ui}`, das den Spaltenmodus verwendet, um einen Gleitkommawert aus dem ListStore auszuwählen.

Wenn Sie so verwirrt sind wie ich, wie man ComboBox ListStores und CellRenderer bearbeitet, lesen Sie https://youtu.be/watch?v=Z5_F-rW2cL8.

Bars (engl. für Balken)

`HAL_Bar` and `HAL_VBar` widgets for horizontal and vertical bars representing floating point values.

Pins

`HAL_Bar` and `HAL_VBar` each have one input REAL HAL pin.

Eigenschaften

Die beiden Balken `HAL_Bar` und `HAL_VBar` haben die folgenden Eigenschaften:

invert

Vertauschen Sie die Richtung von Minimum und Maximum. +
Eine invertierte HBar wächst von rechts nach links, eine invertierte VStabi von oben nach unten.

min, max

Mindest- und Höchstwert des gewünschten Bereichs. Es wird Fehler ausgelöst, wenn der aktuelle Wert außerhalb dieses Bereichs liegt.

show limits (engl. für Grenzen zeigen)

Dient zum Auswählen/Abwählen des Grenzwerttextes auf der Leiste.

zero (Null)

Nullpunkt des Bereichs.
Wenn er innerhalb des Min/Max-Bereichs liegt, wächst der Balken von diesem Wert aus und nicht von der linken (oder rechten) Seite des Widgets.
Nützlich zur Darstellung von Werten, die sowohl positiv als auch negativ sein können.

force_width, force_height

Erzwungene Breite oder Höhe des Widgets. +
Wenn nicht festgelegt, wird die Größe aus der Verpackung oder aus der festen Widgetgröße abgeleitet und die Leiste füllt den gesamten Bereich aus.

text_template

Legt wie bei Label das Textformat für Min/Max/Aktuelle Werte fest. +
Kann verwendet werden, um die Anzeige der Werte auszuschalten.

Wert

Setzt die Balkenanzeige auf den eingegebenen Wert. +
Wird nur zum Testen im GLADE-Editor verwendet. +
Der Wert wird von einem HAL-Pin gesetzt.

target value (engl. für Zielwert)

Setzt die Zielzeile auf den eingegebenen Wert. +
Wird nur zum Testen im GLADE-Editor verwendet. +
Der Wert kann in einer Python-Funktion festgelegt werden.

target_width

Breite der Linie, die den Zielwert markiert.

bg_color

Hintergrund (inaktiv) Farbe des Balken (engl. bar).

target_color

Farbe der Ziellinie.

z0_color, z1_color, z2_color

Farben der verschiedenen Wertzonen. +
Standardwerte sind `grün`, `gelb` und `rot`. +
Für eine Beschreibung der Zonen siehe Eigenschaften von `z*_border`.

z0_border, z1_border

Definieren Sie die Grenzen der Farbzonen. +
Standardmäßig ist nur eine Zone aktiviert. Wenn Sie mehr als eine Zone wünschen, setzen Sie `z0_border` und `z1_border` auf die gewünschten Werte, so dass Zone 0 von 0 bis zur ersten Grenze, Zone 1 von der ersten bis zur zweiten Grenze und Zone 2 von der letzten Grenze bis 1 gefüllt wird. +
Die Ränder werden als Brüche festgelegt. +
Gültige Werte reichen von 0 bis 1.

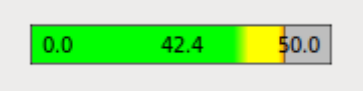


Figure 261. Horizontaler Balken

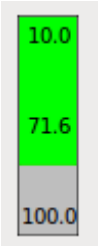


Figure 262. Vertikaler Balken

Meter

`HAL_Meter` is a widget similar to PyVCP meter - it represents a floating point value.

Pins

`HAL_Meter` has one input REAL HAL pin.

Eigenschaften

HAL Meter hat die folgenden Eigenschaften:

min, max

Mindest- und Höchstwert des gewünschten Bereichs.
Es handelt sich nicht um eine Fehlerbedingung, wenn der aktuelle Wert außerhalb dieses Bereichs liegt.

force_size

Erzwungener Durchmesser des Widgets.
Wenn nicht festgelegt, wird die Größe aus der Packung oder aus der festen Widgetgröße abgeleitet, und der Zähler füllt den gesamten verfügbaren Platz unter Berücksichtigung des Seitenverhältnisses.

text_template

Legt, wie bei Label, das Textformat für den aktuellen Wert fest.
Kann verwendet werden, um die Anzeige des Wertes auszuschalten.

label

Großes Etikett über der Metermitte.

Sublabel

Kleines Etikett unter der Mitte des Messgeräts.

bg_color

Hintergrundfarbe des Messgeräts.

z0_color, z1_color, z2_color

Farben der verschiedenen Wertzonen. +
Standardwerte sind `grün`, `gelb` und `rot`. +
Für eine Beschreibung der Zonen siehe Eigenschaften von `z*_border`.

z0_border, z1_border

Definieren Sie die Grenzen der Farbzonen. +
Standardmäßig ist nur eine Zone aktiviert. Wenn Sie mehr als eine Zone wünschen, setzen Sie `z0_border` und `z1_border` auf die gewünschten Werte, so dass Zone 0 vom Minimum bis zum ersten Rand, Zone 1 vom ersten bis zum zweiten Rand und Zone 2 vom letzten Rand bis zum Maximum gefüllt wird. +
Die Ränder werden als Werte im Bereich min-max eingestellt.

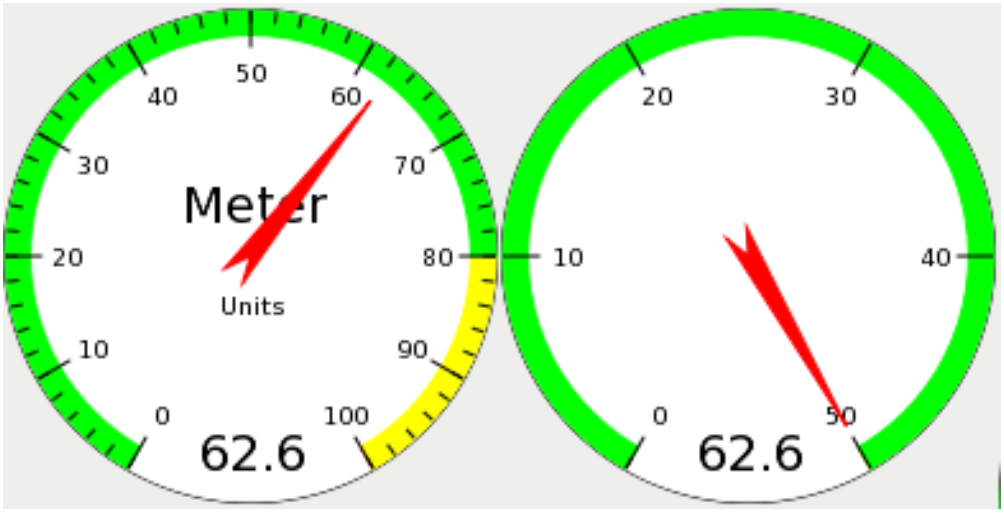


Figure 263. Beispiel HAL-Messgeräte

HAL_Graph

Dieses Widget dient zum Auftragen von Werten über die Zeit.

Gremlin-Werkzeugpfad-Vorschau für NGC-Dateien

Gremlin ist ein Plot-Vorschau-Widget ähnlich dem AXIS-Vorschaufenster. Es setzt eine laufende LinuxCNC-Umgebung wie AXIS oder Touchy voraus. Um sich damit zu verbinden, prüft die INI_FILE_NAME Umgebungsvariable. Gremlin zeigt die aktuelle NGC-Datei an - er überwacht auf Änderungen und lädt die NGC-Datei neu, wenn sich der Dateiname in AXIS/Touchy ändert. Wenn Sie es in einer GladeVCP-Anwendung ausführen, wenn LinuxCNC nicht läuft, könnten Sie einen Traceback bekommen, weil das Gremlin-Widget den LinuxCNC-Status, wie den aktuellen Dateinamen, nicht finden kann.

Pins

Gremlin exportiert keine HAL-Pins.

Eigenschaften

Gremlin hat die folgenden Eigenschaften:

enable_dro

Zeigt den Tropfen auf der Grafik an.
Standard = true.

show_velocity

Dies zeigt die Werkzeuggeschwindigkeit an.
Vorteinstellung = true.

use_commanded

Dies wählt die zu verwendende DRO aus: befohlene oder tatsächliche Werte.
Voreinstellung = true.

metric_units

Dies legt fest was die DRO nutzt: metrische oder imperiale Einheiten.
Voreinstellung = true.

show_rapids

Dies legt fest, dass der Plotter die schnelle Bewegungen zeigt.
Voreinstellung = true.

show_dtg_

Hiermit wird die Anzeige des Restweges auf der DRO ausgewählt.
Voreinstellung = true.

use_relative (relativ anzeigen)

Hiermit wird festgelegt, dass die Anzeige der Werte relativ zu den Koordinaten des Benutzersystems oder der Maschine erfolgen soll.
Voreinstellung = true.

show_live_plot (Live-Plot anzeigen)

Hiermit wird dem Plotter mitgeteilt, ob er zeichnen soll oder nicht.
Voreinstellung = true.

show_limits (engl. für Grenzen zeigen)

Hiermit wird der Plotter angewiesen, die Grenzen der Maschine anzuzeigen.
Voreinstellung = true.

show_lathe_radius (engl. für Drehmaschinenradius anzeigen)

Hiermit wird die DRO Anzeige der X-Achse in Radius oder Durchmesser gewählt, wenn der Drehmaschinenmodus aktiviert ist (wählbar in der INI-Datei mit LATHE = 1).
Voreinstellung = true.

show_extents_option (Ausmaße anzeigen)

Hiermit wird der Plotter angewiesen, die Außenmaße der Maschine anzuzeigen.

Voreinstellung = true.

show_tool (Werkzeug anzeigen)

Hiermit wird der Plotter angewiesen, das Werkzeug zu zeichnen.
Voreinstellung = true.

show_program (Programm zeigen)

Zeigt das G-Code-Programm an.
Standard = True

use_joints_mode

Wird in nicht trivialkins Maschinen (z.B. Robotern) verwendet.
Standard = false.

grid_size

Legt die Größe des Gitters fest (nur in den Ansichten X, Y und Z sichtbar). +
Standardwert ist 0

use_default_controls

Damit wird die Standard-Maussteuerung deaktiviert. +
Dies ist besonders nützlich, wenn Sie einen Touchscreen verwenden, da die Standardsteuerungen nicht gut funktionieren. Sie können programmatisch Steuerelemente mit Python und der Handler-Datei-Technik hinzufügen. +
Standard = true.

view (engl. für Sicht)

Kann einer der Werte `x`, `y`, `y2` , `z`, `z2` , `p` (Perspektive) sein.
Standardmäßig wird die Ansicht `z` verwendet.

enable_dro

Typ = boolesch. +
Ob ein DRO auf dem Plot gezeichnet werden soll oder nicht. +
Voreinstellung = true.

mouse_btn_mode

Typ = Ganzzahl.
Maustastenbehandlung: führt zu verschiedenen Funktionen der Taste:

- 0 = Voreinstellung: Links drehen, Mitte bewegen, rechts zoomen
- 1 = Links zoomen, Mitte verschieben, rechts rotieren
- 2 = links bewegen, mitte drehen, rechts zoomen
- 3 = Links zoomen, Mitte drehen, rechts bewegen
- 4 = Links verschieben, Mitte zoomen, rechts rotieren
- 5 = Links drehen, Mitte zoomen, rechts bewegen
- 6 = Bewegung nach links, mittlerer Zoom, rechter Zoom

Modus 6 wird für Plasmas und Drehbänke empfohlen, da für diese Maschinen keine Rotation erforderlich ist.

Direkte Programmsteuerung

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie GObject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property('view','P')
[widget name].set_property('metric_units',False)
[widget name].set_property('use_default_controls',False)
[widget name].set_property('enable_dro' False)
[widget name].set_property('show_program', False)
[widget name].set_property('show_limits', False)
[widget name].set_property('show_extents_option', False)
[widget name].set_property('show_live_plot', False)
[widget name].set_property('show_tool', False)
[widget name].set_property('show_lathe_radius',True)
[widget name].set_property('show_dtg',True)
[widget name].set_property('show_velocity',False)
[widget name].set_property('mouse_btn_mode', 4)
```

Es gibt Python-Methoden:

```
[widget name].show_offsets = True
[widget name].workpiece_opacity = .25
[widget name].grid_size = .75
[widget name].select_fire(event.x,event.y)
[widget name].select_prime(event.x,event.y)
[widget name].start_continuous_zoom(event.y)
[widget name].set_mouse_start(0,0)
[widget name].gremlin.zoom_in()
[widget name].gremlin.zoom_out()
[widget name].get_zoom_distance()
[widget name].set_zoom_distance(dist)
[widget name].clear_live_plotter()
[widget name].rotate_view(x,y)
[widget name].pan(x,y)
```

Hinweise

- Wenn Sie alle Plot-Optionen auf false, show_offsets aber auf true setzen, erhalten Sie eine Seite mit Offsets anstelle einer grafischen Darstellung.
- Es ist viel benutzerfreundlicher, wenn Sie die Zoomdistanz ermitteln, bevor Sie die Ansicht ändern, und dann die Zoomdistanz zurücksetzen.
- wenn Sie ein Element in der Vorschau auswählen, wird das ausgewählte Element als Rotationsmittelpunkt verwendet

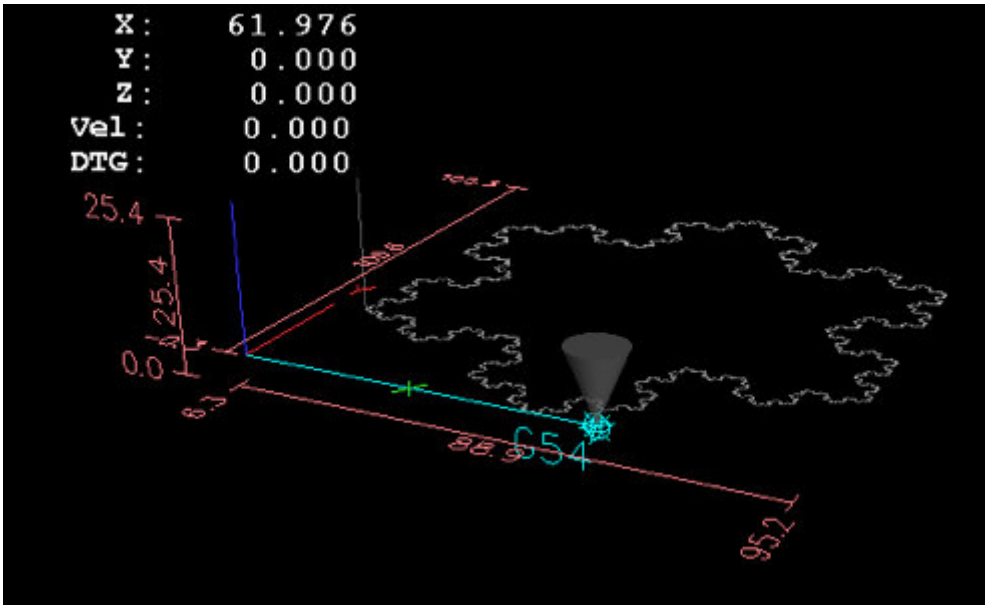


Figure 264. Gremlin Beispiel

HAL_Offset

Das `HAL_Offset`-Widget wird verwendet, um den Offset einer einzelnen Achse anzuzeigen.

Eigenschaften

`HAL_Offset` hat die folgenden Eigenschaften:

display_units_mm

Anzeige in metrischen Einheiten.

joint_number

Dient zur Auswahl, welche Achse (technisch gesehen welche Gelenk) angezeigt wird.
Auf einer Trivialkins-Maschine (Fräse, Drehmaschine, Oberfräse) sind Achse und Gelenknummer gleich:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W
```

mm_text_template

Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

imperial_text_template

Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

reference_type

```
0:G5x 1:Werkzeug 2:G92 3:Drehung um Z
```

DRO-Widget

Das DRO-Widget wird verwendet, um die aktuelle Achsenposition anzuzeigen.

Eigenschaften

Es hat die folgenden Eigenschaften:

display_units_mm

Dient zum Umschalten der Anzeigeeinheiten zwischen metrisch und imperial. Standard ist False.

actual (engl. für tatsächlich)

Wählen Sie die tatsächliche Position (Rückmeldung, engl. feedback) oder die Sollposition aus. Der Standardwert ist True.

Durchmesser

Durchmesser für eine Drehmaschine anzeigen. Standard ist False.

mm_text_template

Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen. Standard ist "%10.3f".

imperial_text_template

Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen. Standard ist "%9.4f".

joint_number

Dient zur Auswahl, welche Achse (technisch gesehen welches Gelenk) angezeigt wird. Standard ist 0. Auf einer Trivialkins-Maschine (Fräsmaschine, Drehmaschine, Oberfräse) sind die Achsen im Vergleich zur Gelenknummer:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W +
```

reference_type

- 0 = `absolute` (Maschinen-Ursprung).
- 1 = `relative` (zum aktuellen Ursprung der Benutzerkoordinate - G5x).
- 2 = `distance-to-go` (engl. für verbleibender Weg) (relativ zum aktuellen Ursprung der Benutzerkoordinate). Standardeinstellung ist 0.

font_family

Geben Sie die Schriftfamilie an, z. B. mono. Standardmäßig ist sans eingestellt. Wenn die Schriftart nicht vorhanden ist, wird die aktuelle Systemschriftart verwendet. Standardwert ist sans.

font_size

Geben Sie die Größe der Schrift zwischen 8 und 96 an. Standard ist 26.

font_weight

Geben Sie die Stärke der Schrift an. Wählen Sie zwischen ligher (engl. für heller), normal, bold (engl. für fett) und bolder (engl. für fetter). Standard ist bold (engl. für fett).

unhomed_color

Die Farbe des Textes, wenn er nicht beherbergt ist, wird als Gdk.RGBA-Farbe angegeben. Standard ist rot, Gdk.RGBA(red=1.000000, green=0.000000, blue=0.000000, alpha=1.000000), jeweils englisch für rot, grün und blau

homed_color

Die Farbe des Textes, wenn Maschine nicht am Referenzpunkt, wird als Gdk.RGBA-Farbe angegeben. Standard ist rot, Gdk.RGBA(red=0.000000, green=0.501961, blue=0.000000, alpha=1.000000), jeweils englisch für rot, grün und blau

Hinweise

- Wenn die Anzeige rechtsbündig sein soll, stellen Sie die horizontale Ausrichtung auf `End` (engl. für Ende).
- Der Hintergrund des Widgets ist eigentlich durchsichtig - wenn Sie es also über einem Bild platzieren, werden die DRO-Zahlen oben ohne Hintergrund angezeigt. Es gibt eine spezielle Technik, um dies zu erreichen. Siehe die animierten Funktionsdiagramme unten.
- Das DRO Widget ist ein modifiziertes gtk Label Widget. Als solches kann dem DRO Widget viel von dem getan werden, was mit einem gtk-Label gemacht werden kann.
- Die Schrifteigenschaften können auch über ein CSS-Stylesheet festgelegt werden, das die höchste Priorität hat und die durch GObject-Eigenschaften festgelegten Werte außer Kraft setzt.

Direkte Programmsteuerung

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwendung von GObject zur Einstellung der oben aufgeführten Eigenschaften

```
[widget name].set_property("display_units_mm", True)
[widget name].set_property("actual", True)
[widget name].set_property("diameter", True)
[widget name].set_property("mm_text_template", "%10.3f")
[widget name].set_property("imperial_text_template", "%9.4f")
[widget name].set_property("joint_number", 3)
[widget name].set_property("reference_type", 3)
[widget name].set_property("font_family", "mono")
[widget name].set_property("font_size", 30)
[widget name].set_property("font_weight", "bold")
```

```
# es ist einfacher, Farben durch den Aufruf einer Funktion zu lesen:
def str_to_rgba(color):
  c = Gdk.RGBA()
  c.parse(color)
  return c
```

```
[widget name].set_property("unhomed_color", str_to_rgba("magenta"))
[widget name].set_property("homed_color", str_to_rgba("cyan"))
```

Verwendung eines CSS-Stylesheets zum Festlegen von Schrifteigenschaften

Farben können in einem von mehreren Formaten angegeben werden, die alle dieselbe Farbe angeben, nämlich Rot, *#ff0000, *rgb(255,0,0) oder rgba(255,0,0,255).

Die Farben können entweder gemeinsam referenziert werden:

```
.dro_unhomed {color: magenta}
.dro_homed {color: cyan}
```

oder einzeln nach Widget-Namen:

```
#[widget name].dro_unhomed {color: magenta}
#[widget name].dro_homed {color: cyan}
```

Die anderen Stileigenschaften müssen über den Widgetnamen referenziert werden:

```
#[widget name], #[widget name], #[widget name] {
    font-family: mono;
    font-size: 60px;
    font-weight: lighter;
}
```

Es gibt zwei Python-Methoden

```
[widget name].set_dro_inch()
[widget name].set_dro_metric()
```

Combi_DRO Widget

Das `Combi_DRO`-Widget wird verwendet, um die aktuelle, die relative Achsenposition und die zu fahrende Strecke in einem DRO anzuzeigen.
Durch Klicken auf das DRO wird die Reihenfolge der DRO umgeschaltet.
Im relativen Modus wird das aktuelle Koordinatensystem angezeigt.

Eigenschaften

`Combi_DRO` hat die folgenden Eigenschaften:

joint_number

Dient zur Auswahl, welche Achse (technisch gesehen welche Gelenk) angezeigt wird.
Auf einer Trivialkins-Maschine (Fräse, Drehmaschine, Oberfräse) sind Achse und Gelenknummer gleich:

```
0:X 1:Y 2:Z usw.
```

actual (engl. für tatsächlich)

Wählen Sie die tatsächliche (Rückmeldung) oder die befohlene Position.

metric_units

Dient zum Umschalten der Anzeigeeinheiten zwischen metrisch und imperial.

auto_units

Die Einheiten werden zwischen metrisch und imperial umgeschaltet, je nachdem, ob der aktive G-Code G20 oder G21 ist. +
Voreinstellung ist TRUE.

Durchmesser

Ob die Position als Durchmesser oder Radius angezeigt werden soll. +
Im Durchmessermodus zeigt die Positionsanzeige den Gelenkwert multipliziert mit 2 an.

mm_text_template

Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.
Standard ist "%10.3f".

imperial_text_template

Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.
Standard ist "%9.4f".

homed_color

Die Vordergrundfarbe der DRO-Nummern, wenn das Gelenk referenziert ist.
Standard ist grün.

unhomed_color

Die Vordergrundfarbe der DRO-Nummern, wenn das Gelenk nicht referenziert ist.
Standard ist rot.

abs_color

Die Hintergrundfarbe der DRO, wenn die Haupt-DRO absolute Koordinaten anzeigt.
Standard ist blau.

rel_color

Die Hintergrundfarbe des DRO, wenn das Haupt-DRO relative Koordinaten anzeigt.
Standard ist schwarz.

dtg_color

Die Hintergrundfarbe der DRO, wenn die Haupt-DRO die verbleibende Entfernung anzeigt.
Standard ist gelb.

font_size

Die Schriftgröße der großen Zahlen, die kleinen Zahlen werden 2,5 mal kleiner sein.

Der Wert muss eine ganze Zahl im Bereich von 8 bis 96 sein.
Standardwert ist 25.

toggle_readout

Ein linker Mausklick schaltet die DRO-Anzeige zwischen den verschiedenen Modi um ["Rel", "Abs", "DTG"].
Durch Deaktivieren des Kontrollkästchens können Sie dieses Verhalten abschalten. Das Umschalten kann immer noch mit `[Widgetname].toggle_readout()` durchgeführt werden.
Der Wert muss boolesch sein.
Voreinstellung ist TRUE.

cycle_time

Die Zeit, die ein DRO zwischen zwei Abfragen wartet.
Diese Einstellung sollte nur geändert werden, wenn Sie mehr als 5 DROs gleichzeitig verwenden, z.B. bei einer 6-Achsen-Konfiguration, um zu vermeiden, dass die DRO die Hauptanwendung zu sehr verlangsamt.
Der Wert muss eine ganze Zahl im Bereich von 100 bis 1000 sein. FIXME unit=ms ?
Voreinstellung ist 150.

Direkte Programmsteuerung

Verwenden Sie GObject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property(property, value)
```

Es gibt mehrere Python-Methoden zur Steuerung des Widgets:

- `[Widgetname].set_to_inch(state)`
  Legt fest, dass die DRO imperiale Einheiten anzeigt.
  `state` = boolesch (True oder False)
  Voreinstellung ist FIXME.
- `[Widgetname].set_auto_units(state)`
  Wenn True, ändert die DRO die Einheiten entsprechend dem aktiven G-Code (G20 / G21).
  `state` = boolean (Wahr oder Falsch)
  Standard ist True.
- `[Name des Widgets].set_to_diameter(state)`
  Wenn True, zeigt die DRO den Durchmesser und nicht den Radius an, d. h. den Achsenwert multipliziert mit 2 (speziell für Drehmaschinen erforderlich).
  `state` = boolean (Wahr oder Falsch)
  Der Standardwert ist Falsch.
- `[Widgetname].toggle_readout()`
  Schaltet die Reihenfolge des DRO im Widget um.
- *[Widget-Name].change_axisletter(Buchstabe)*
  Ändert den automatisch angegebenen Achsenbuchstaben.
  Sehr nützlich, um eine Drehmaschine DRO von *X* auf *R* oder *D* zu ändern.
  letter = Zeichenfolge

- `[Widgetname].get_order()`
  Gibt die Reihenfolge der DRO im Widget zurück, die hauptsächlich dazu dient, sie konsistent zu halten.
  Die Reihenfolge wird auch mit dem Klicksignal übertragen.
  Gibt eine Liste mit der Reihenfolge zurück.
- `[Name des Widgets].set_order(order)`
  Legt die Reihenfolge der DRO fest, die hauptsächlich verwendet wird, um sie konsistent zu halten.
  `order` = Listenobjekt, muss eines der folgenden sein:
  - `["Rel", "Abs", "DTG"]` (Standard)
  - `["DTG", "Rel", "Abs"]`
  - `["Abs", "DTG", "Rel"]`
- `[Widgetname].get_position()`
  Liefert die Position des DRO als eine Liste von Floats.
  Die Reihenfolge ist unabhängig von der auf dem DRO angezeigten Reihenfolge und wird als `[Absolut , relativ , DTG]` angegeben.
  - `Absolut` = die Maschinenkoordinaten, abhängig von der tatsächlichen Eigenschaft, die eine tatsächliche oder befohlene Position ergibt.
  - `Relativ` = sind die Koordinaten des aktuellen Koordinatensystems.
  - `DTG` = die zu gehende Strecke.
    Wird meistens 0 sein, da diese Funktion aufgrund von Zeitverzögerungen nicht verwendet werden sollte, während sich die Maschine bewegt.

Das Widget sendet die folgenden Signale aus:

- `clicked`
  Dieses Signal wird ausgesendet, wenn der Benutzer auf das Combi_DRO-Widget geklickt hat.
  Es wird die folgenden Daten senden:
  - `widget` = Widget-Objekt
    Das Widget-Objekt, welches das Signal sendet.
  - `joint_number` = ganze Zahl
    Die gemeinsame Nummer des DRO, wobei *0:X 1:Y 2:Z etc*.
  - `order` = Listenobjekt +
    Die Reihenfolge des DRO in diesem Widget. +
    Die Reihenfolge kann verwendet werden, um andere Combi_DRO-Widgets mit `[widget name].set_order(order)` auf die gleiche Reihenfolge zu setzen.
- `units_changed` +
  Dieses Signal wird ausgegeben, wenn die DRO-Einheiten gewechselt werden. +
  Es sendet die folgenden Daten:
  - `widget` = Widget-Objekt
    Das Widget-Objekt, welches das Signal sendet.
  - `metric_units` = boolesch +
    True, wenn die DRO metrische Einheiten anzeigt, False, wenn die Anzeige imperial ist.

- `system_changed`+
  Dieses Signal wird ausgegeben, wenn die DRO-Einheiten gewechselt werden. +
  Es sendet die folgenden Daten:
  - `widget` = Widget-Objekt
    Das Widget-Objekt, welches das Signal sendet.
  - `system` = Zeichenfolge +
    Das eigentliche Koordinatensystem. Wird einer von G54 G55 G56 G57 G58 G59 G59.1 G59.2 G59.3 oder Rel sein, wenn überhaupt nicht ausgewählt wurde, was nur in Glade passieren wird, wenn kein LinuxCNC läuft.

Es gibt einige Informationen, die Sie über Befehle erhalten können und die für Sie von Interesse sein könnten:

- `[Widgetname].system`
  Das eigentliche System, wie im Signal system_changed erwähnt.
- `[Widget-Name].homed` +
  True, wenn das Gelenk referenziert ist.
- `[Widgetname].machine_units` +
  0 wenn imperial, 1 wenn metrisch.

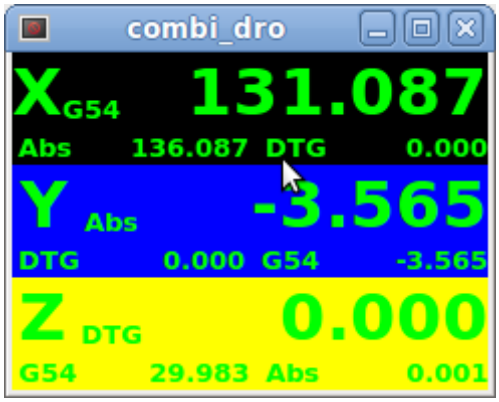


Figure 265. Beispiel: Drei Combi_DRO in einem Fenster

```
X = Relativer Modus +
Y = Absoluter Modus +
Z = DTG-Modus
```

IconView (Dateiauswahl)

Dies ist ein Touchscreen-freundliches Widget zur Auswahl einer Datei und zum Wechseln von Verzeichnissen.

Eigenschaften

Das `IconView`-Widget hat die folgenden Eigenschaften:

icon_size

Legt die Größe des angezeigten Symbols fest. +

Erlaubte Werte sind Ganzzahlen im Bereich von 12 bis 96. +
Voreinstellung ist 48.

start_dir

Legt das Verzeichnis fest, in dem das Widget beim ersten Mal angezeigt wird.
Muss ein String sein, der einen gültigen Verzeichnispfad enthält.
Standard ist "/".

jump_to_dir

Legt das Verzeichnis fest, in das gesprungen werden soll und das durch die entsprechende Schaltfläche in der unteren Schaltflächenliste ausgewählt wird (die 5. Schaltfläche von links).
Muss ein String sein, der einen gültigen Verzeichnispfad enthält.
Voreinstellung ist "\~".

filetypes

Legt den Dateifilter für die anzuzeigenden Objekte fest.
Muss eine Zeichenkette sein, die eine durch Komma getrennte Liste von Erweiterungen enthält, die angezeigt werden sollen.
Standard ist "ngc,py".

sortorder

Legt die Sortierreihenfolge des angezeigten Symbols fest. +
Muss ein ganzzahliger Wert von 0 bis 3 sein, mit den entsprechenden Konstanten:

- 0 = ASCENDING (engl. für aufsteigend) (nach Dateinamen sortiert)
- 1 = DESCENDING (engl. für absteigend) (Sortierung nach Dateinamen)
- 2 = FOLDERFIRST (zuerst die Ordner, dann die Dateien anzeigen), Standard
- 3 = FILEFIRST (zuerst die Dateien, dann die Ordner anzeigen)

Direkte Programmsteuerung

Verwenden Sie GObject, um die oben aufgeführten Eigenschaften einzustellen:

```
[Widgetname].set_property(Eigenschaft,Wert)
```

Es gibt Python-Methoden zur Steuerung des Widgets:

- `[Widgetname].show_buttonbox(state)` +
  Bei False wird das untere Schaltflächenfeld ausgeblendet. +
  Dies ist hilfreich in benutzerdefinierten Bildschirmen, mit speziellen Schaltflächen-Layouts, um das Layout der GUI nicht zu verändern. Ein gutes Beispiel dafür ist GMOCCAPY. +
  `state` = boolescher Wert (True oder False). +
  Voreinstellung ist True.
- `[Widgetname].show_filelabel(state)`+
  Bei True wird das Dateilabel (zwischen dem IconView-Fenster und dem unteren Schaltflächenfeld) angezeigt. +

Das Ausblenden dieses Labels kann Platz sparen, aber das Anzeigen ist sehr nützlich für die Fehlersuche. +
`state` = boolescher Wert (True oder False). +
Voreinstellung ist True.

- `[Widgetname].set_icon_size(iconsize)` +
  Setzt die Größe des Icons. +
  Muss eine ganze Zahl im Bereich von 12 bis 96 sein. +
  Voreinstellung = 48.
- `[Widgetname].set_directory(Verzeichnis)` +
  Ermöglicht das Setzen eines anzuzeigenden Verzeichnisses. +
  `Verzeichnis` = String (ein gültiger Dateipfad).
- `[Widgetname].set_filetypes(Dateitypen)` +
  Legt den zu verwendenden Dateifilter fest. +
  Es werden nur Dateien mit den angegebenen Erweiterungen angezeigt. +
  `Dateitypen` = String mit einer durch Komma getrennten Liste von Erweiterungen. +
  Voreinstellung = "ngc,py".
- `[Widgetname].get_selected()` +
  Gibt den Pfad der ausgewählten Datei zurück, oder `None`, wenn ein Verzeichnis ausgewählt wurde.
- `[Widgetname].refresh_filelist()` +
  Aktualisiert die Dateiliste. +
  Wird benötigt, wenn Sie eine Datei hinzufügen, ohne das Verzeichnis zu ändern.

Wenn der Button ausgeblendet wurde, können Sie die Funktionen dieses Buttons über die angeklickten Signale wie folgt erreichen:

```
[widget name].btn_home.emit("clicked")
[widget name].btn_jump_to.emit("clicked")
[widget name].btn_sel_prev.emit("clicked")
[widget name].btn_sel_next.emit("clicked")
[widget name].btn_get_selected.emit("clicked")
[widget name].btn_dir_up.emit("clicked")
[widget name].btn_exit.emit("clicked")
```

Signale

Das Widget sendet die folgenden Signale aus:

- `selected` (engl. für ausgewählt) +
  Dieses Signal wird ausgegeben, wenn der Benutzer ein Symbol auswählt. +
  Es gibt einen String zurück, der einen Dateipfad enthält, wenn eine Datei ausgewählt wurde, oder `None`, wenn ein Verzeichnis ausgewählt wurde.
- `empfindlich` +
  Dieses Signal wird ausgegeben, wenn die Schaltflächen ihren Zustand von sensitiv zu nicht sensitiv oder umgekehrt ändern. +
  Dieses Signal ist nützlich, um die umgebende GUI mit der Schaltfläche des Widgets synchronisiert zu halten. Siehe GMOCCAPY als Beispiel. +

Es gibt den **Buttonnamen** und den neuen **Zustand** (engl. state) zurück:

- Der Name der Schaltfläche ist einer der folgenden: `btn_home`, `btn_dir_up`, `btn_sel_prev`, `btn_sel_next`, `btn_jump_to` oder `btn_select`.
- `state` ist ein boolescher Wert und wird True oder False sein.

- `exit` +
  Dieses Signal wird ausgegeben, wenn der Exit-Button gedrückt wurde, um die IconView zu schließen. +
  Meistens benötigt, wenn die Anwendung als eigenständige Anwendung gestartet wird.

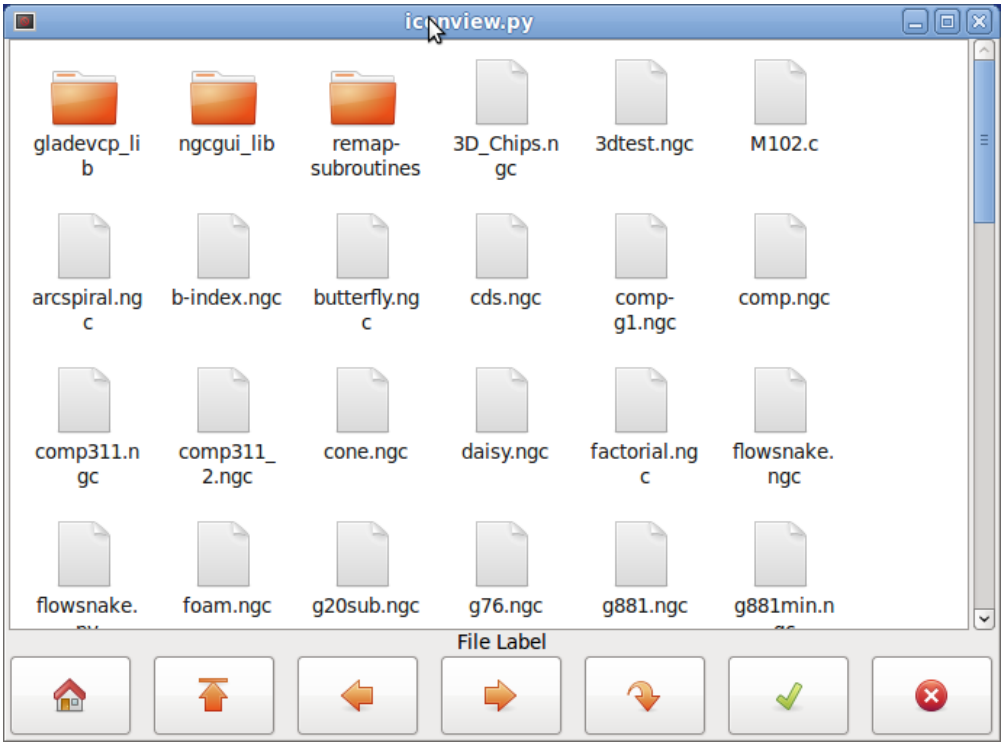


Figure 266. Iconview Beispiel

Rechner-Widget

Dies ist ein einfaches Taschenrechner-Widget, das für numerische Eingaben verwendet werden kann. +
Sie können die Anzeige voreinstellen und das Ergebnis oder den voreingestellten Wert abrufen.

Eigenschaften

`Rechner` (engl. calculator) hat folgende Eigenschaften:

is_editable

Damit kann die Eingabeanzeige über eine Tastatur eingegeben werden.

font

Hier können Sie die Schriftart für die Anzeige einstellen.

Direkte Programmsteuerung

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie goobject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property("is_editable",True)
[widget name].set_property("font","sans 25")
```

Es gibt Python-Methoden:

- `[Widgetname].set_value(2.5)`
  Damit ist die Anzeige voreingestellt und wird aufgezeichnet.
- `[Widgetname].set_font("sans 25")`
- `[Widgetname].set_editable(True)`
- `[Widgetname].get_value()`
  Gibt den berechneten Wert zurück - eine Fließkommazahl.
- `[Widgetname].set_editable(True)`
- `[Widgetname].get_preset_value()`
  Gibt den aufgezeichneten Wert zurück: eine Fließkommazahl.

Werkzeugeditor-Widget (engl. tooleditor widget)

Dies ist ein "Werkzeug-Editor"-Widget zum Anzeigen und Ändern einer Werkzeugdatei. +
Im Drehmaschinenmodus werden Verschleißkorrekturen und Werkzeugkorrekturen separat angezeigt. +
Verschleißkorrekturen werden durch Werkzeugnummern über 10000 (Fanuc-Stil) gekennzeichnet. +
Es überprüft die aktuelle Datei einmal pro Sekunde, um zu sehen, ob LinuxCNC es aktualisiert.

NOTE	LinuxCNC erfordert Mapping von Werkzeug-Aufrufe, um Verschleiß-Offsets anzuwenden.

Eigenschaften

`tooleditor` hat die folgenden Eigenschaften:

font

Zu verwendende Anzeige-Schriftart

hide_columns

Dadurch werden die angegebenen Spalten ausgeblendet. +
Die Spalten werden (in dieser Reihenfolge) wie folgt bezeichnet: `s,t,p,x,y,z,a,b,c,u,v,w,d,i,j,q`. +
Sie können eine beliebige Anzahl von Spalten ausblenden, einschließlich der Auswahl und der Kommentare.

lathe_display_type

Format der Drehmaschine anzeigen

Direkte Programmsteuerung

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie goobject, um die oben aufgeführten Eigenschaften einzustellen:

```
[Widgetname].set_properties('hide_columns','uvwijq')
```

Dadurch würden die Spalten uvwij und q ausgeblendet und alle anderen angezeigt.

Es gibt Python-Methoden:

- `[Widgetname].set_visible("ijq",False)`
  Blendet die Spalten ij und Q aus und belässt den Rest so, wie er war.
- `[Widgetname].set_filename(pfad_zur_datei)`
  Setzt und lädt die Werkzeugdatei.
- `[Widgetname].reload(None)`
  Lädt die aktuelle Werkzeugdatei neu.
- `[Widgetname].set_font('sans 16,tab='1')` +
  Setzt die (Pango)-Schriftart für die Registerkarte, den Spaltentitel und die Werkzeugdaten. +
  Die `all_offsets`, `wear_offsets`, `tool_offsets` können gleichzeitig gesetzt werden, indem man 1, 2 und/oder 3 an den Tab-String anhängt. +
  Standardmäßig sind alle Registerkarten eingestellt.
- `[Widgetname].set_title_font('sans 16,tab='1')` +
  Setzt die (Pango)-Schriftart nur für die Spaltentitel. +
  Die `all_offsets`, `wear_offsets`, `tool_offsets` können gleichzeitig gesetzt werden, indem man 1, 2 und/oder 3 an den Tab-String anhängt. +
  Standardmäßig sind alle Tabulatoren gesetzt.
- `[Widgetname].set_tab_font('sans 16,tab='1')` +
  Setzt die (Pango)-Schriftart nur auf den Registerkarten. +
  Die `all_offsets`, `wear_offsets`, `tool_offsets` können gleichzeitig gesetzt werden, indem man 1, 2 und/oder 3 an den Tab-String anhängt. +
  Standardmäßig sind alle Tabs gesetzt.
- `[Widgetname].set_col_visible("abcUVW", False, tab='1')` +
  Dies würde die abcuvw-Spalten auf der Registerkarte 1 (all_offsets) ausblenden (False)
- `[widget name].set_lathe_display(value)` +
  Blendet die Verschleiß- und Werkzeugkorrekturtabellen für Drehbänke ein oder aus
- `[Widgetname].get_toolinfo(toolnum)` +
  Gibt das Werkzeuginformationsfeld der angeforderten Werkzeugnummer oder des aktuellen Werkzeugs zurück, wenn keine Werkzeugnummer angegeben ist. +
  Gibt None zurück, wenn das Werkzeug nicht in der Tabelle gefunden wurde oder wenn es kein

aktuelles Werkzeug gibt.

- `[Widgetname].hide_buttonbox(self, True)` +
  Komfortable Methode zum Ausblenden von Schaltflächen. +
  Sie müssen diese Methode nach show_all() aufrufen.
- `[Widgetname].get_selected_tool()` +
  Gibt die vom Benutzer ausgewählte (hervorgehobene) Werkzeugnummer zurück.
- `[Widgetname].set_selected_tool(toolnumber)` +
  Wählt das gewünschte Werkzeug aus (markiert es).

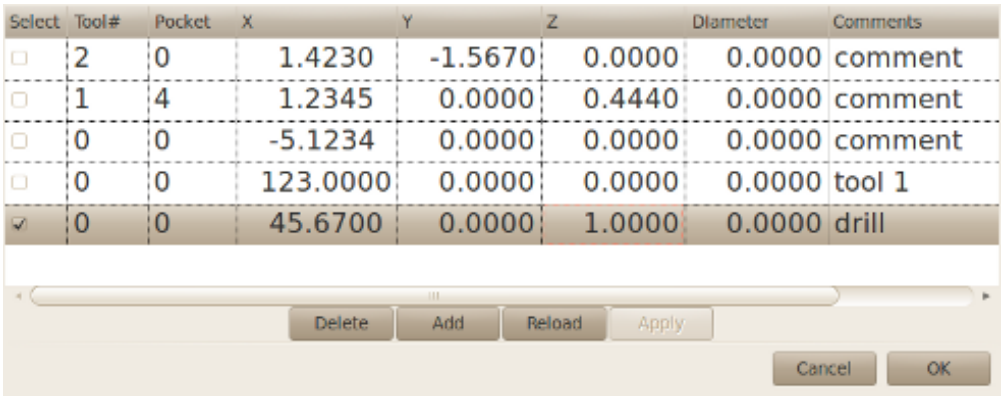


Figure 267. Werkzeug-Editor (engl. tooleditor) Beispiel

Offset-Seite

Das Widget `Offsetpage` dient der Anzeige/Bearbeitung der Offsets aller Achsen.
Es hat praktische Schaltflächen für die Nullstellung von G92- und Rotation-Around-Z-Offsets.
Sie können den Bearbeitungsmodus nur auswählen, wenn die Maschine eingeschaltet und im Leerlauf ist.
Zu diesem Zeitpunkt können Sie die Offsets in der Tabelle direkt bearbeiten. Heben Sie die Auswahl der Schaltfläche "Bearbeiten" auf, damit die Offset-Seite die Änderungen wiedergeben kann.

Eigenschaften

Es hat die folgenden Eigenschaften:

display_units_mm

Anzeige in metrischen Einheiten

hide_columns

Eine Liste der auszublendenden Spalten ohne Leerzeichen. Die Spalten werden (in dieser Reihenfolge) wie folgt bezeichnet: `xyzabcuvwt`.
Sie können jede der Spalten ausblenden.

hide_rows

Eine Liste der auszublendenden Zeilen ohne Leerzeichen.
Die Zeilen werden (der Reihe nach) wie folgt bezeichnet: `0123456789abc`.
Sie können jede der Zeilen ausblenden.

font

Legt Art und Größe der Schriftart fest.

highlight_color

Beim Bearbeiten ist dies die Hervorhebungsfarbe.

foreground_color

Wenn `OffsetPage` ein aktives Benutzerkoordinatensystem erkennt, wird diese Farbe für den Text verwendet.

mm_text_template

Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

imperial_text_template

Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

Direkte Programmsteuerung

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie goobject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property("highlight_color",gdk.Color('blue'))
[widget name].set_property("foreground_color",gdk.Color('black'))
[widget name].set_property("hide_columns","xyzabcuvwt")
[widget name].set_property("hide_rows","123456789abc")
[widget name].set_property("font","sans 25")
```

Es gibt Python-Methoden zur Steuerung des Widgets:

- `[widget name].set_filename("../../../configs/sim/gscreen/gscreen_custom/sim.var")`
- `[widget name].set_col_visible("Yabuvw",False)`
- `[widget name].set_row_visible("456789abc",False)`
- `[widget name].set_to_mm()`
- `[widget name].set_to_inch()`
- `[widget name].hide_button_box(True)`
- `[widget name].set_font("sans 20")`
- `[widget name].set_highlight_color("violet")`
- `[widget name].set_foreground_color("yellow")`
- `[Widgetname].mark_active("G55")`
  Ermöglicht es Ihnen, eine Zeile direkt zu markieren, z.B. wenn Sie Ihre eigenen Navigationskontrollen verwenden möchten. Siehe das Kapitel über GMOCCAPY.

- `[Widgetname].selection_mask = ("Werkzeug", "Rot", "G5x")` +
  Diese Zeilen sind im Bearbeitungsmodus NICHT auswählbar.
- `[Widgetname].set_names([['G54','Default'],["G55", "Vice1"],['Rot','Rotational']])` +
  Damit können Sie den Text der Spalte "T" in jeder beliebigen Zeile festlegen. +
  Es handelt sich um eine Liste von Offset-Namen/Benutzernamen-Paaren. +
  Der Standardtext ist derselbe wie der Offsetname.
- [Widgetname].get_names()` +
  Diese Funktion gibt eine Liste von Paaren aus Zeilen-Schlüsselwort/Benutzername zurück. +
  Die Spalte mit den Benutzernamen ist editierbar, so dass das Speichern dieser Liste benutzerfreundlich ist. +
  Siehe `set_names` oben.

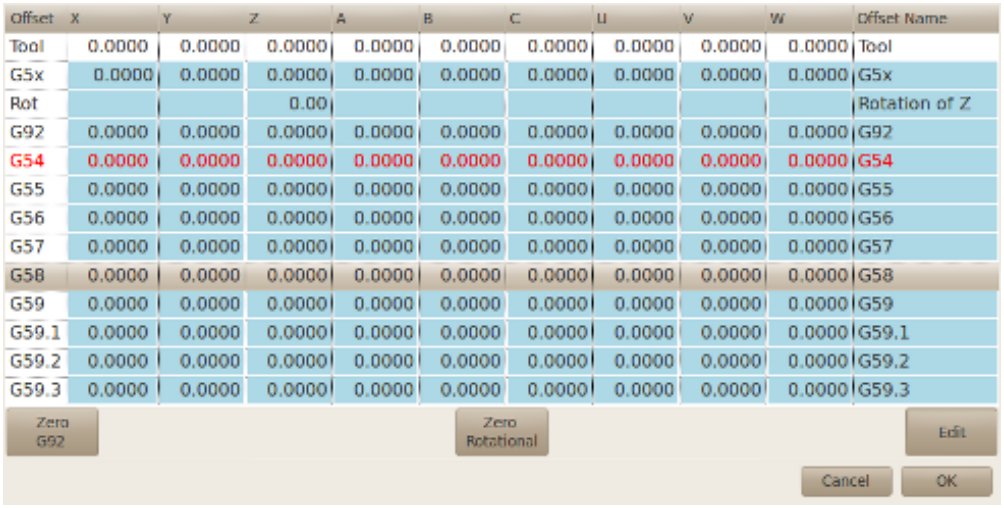


Figure 268. Beispiel für eine Offset-Seite

HAL_sourceview-Widget

Dies ist für die Anzeige und einfache Bearbeitung von G-Code. Es sucht nach `.ngc` -Hervorhebungsspezifikationen in `~/share/gtksourceview-4/language-specs/`. Die aktuell laufende Zeile wird hervorgehoben.

Mit externem Python-Glue-Code kann dies:

- Nach Text suchen, Änderungen rückgängig machen und Wiederherstellen,
- für die Auswahl von Programmzeilen genutzt werden.

Direkte Programmsteuerung

Es gibt Python-Methoden zur Steuerung des Widgets:

- `[widget name].redo()` +
  Wiederholung einer Ebene von Änderungen.
- [widget name].undo()` +
  Rückgängig machen einer Ebene von Änderungen

- `[Widgetname].text_search(direction=True,mixed_case=True,text='G92')` +
  Sucht vorwärts (direction = True) oder rückwärts, +
  Sucht mit gemischter Groß- und Kleinschreibung (mixed_case = True) oder exakter Übereinstimmung
- `[Widgetname].set_line_number(Zeilennummer)` +
  Setzt die hervorzuhebende Zeile. +
  Verwendet die Zeilennummern der Quellansicht.
- `[Widgetname].get_line_number()` +
  Gibt die aktuell hervorgehobene Zeile zurück.
- [Widget-Name].line_up()` +
  Verschiebt die markierte Zeile um eine Zeile nach oben.
- `[Name des Widgets].line_down()`
  Verschiebt die markierte Zeile um eine Zeile nach unten.
- `[Widgetname].load_file('Dateiname')`
  Lädt eine Datei.
  Die Verwendung von None (keine Zeichenkette für den Dateinamen) lädt das gleiche Programm erneut.
- *[Widget-Name].get_filename()*
  FIXME Beschreibung

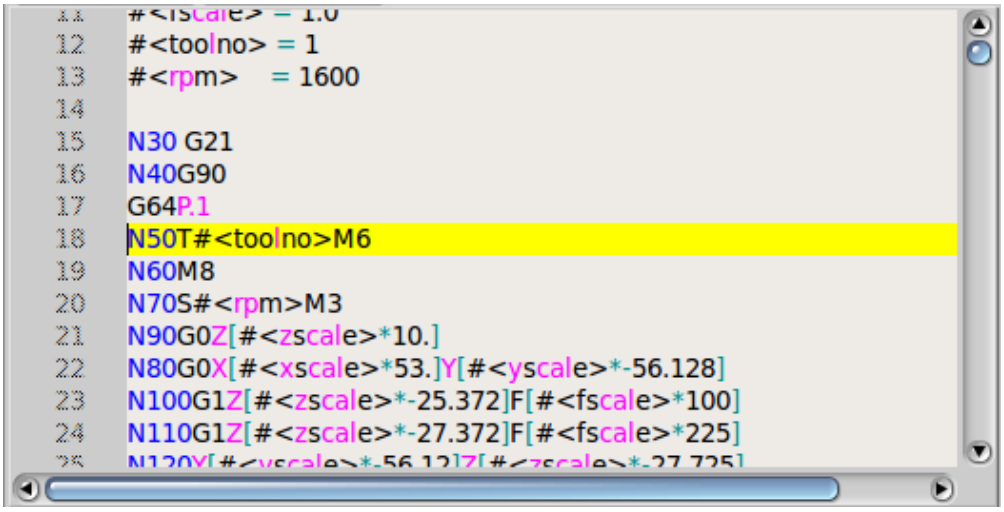


Figure 269. Quellen-Ansicht (engl. sourceview) Beispiel

MDI-Geschichte

Dient zur Anzeige und Eingabe von MDI-Codes.
Sie wird automatisch ausgegraut, wenn MDI nicht verfügbar ist, z. B. bei Notaus und laufenden Programmen.

Eigenschaften

font_size_tree

Ganzzahliger Wert zwischen 8 und 96.
Ändert die Standardschriftgröße der Baumansicht auf den ausgewählten Wert.

font_size_entry

Ganzzahliger Wert zwischen 8 und 96. +
Ändert die Standardschriftgröße der Baumansicht auf den ausgewählten Wert.

use_double_click

Boolean, True aktiviert die Maus-Doppelklick-Funktion und ein Doppelklick auf einen Eintrag sendet diesen Befehl.
Es wird nicht empfohlen, diese Funktion auf echten Maschinen zu verwenden, da ein Doppelklick auf einen falschen Eintrag zu gefährlichen Situationen führen kann.

Direkte Programmsteuerung

Verwenden Sie goobject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property("font_size_tree",10)
[widget name].set_property("font_size_entry",20)
[widget name].set_property("use_double_click",False)
```

Animierte Funktionsdiagramme: HAL-Widgets in einer Bitmap

Für einige Anwendungen kann es wünschenswert sein, ein Hintergrundbild zu haben - wie ein Funktionsdiagramm - und Widgets an geeigneten Stellen in diesem Bild zu positionieren. Eine gute Kombination ist das Setzen eines Bitmap-Hintergrundbildes, z.B. aus einer .png-Datei, das Festlegen der Größe des GladeVCP-Fensters und die Verwendung des Glade Fixed-Widgets zur Positionierung von Widgets auf diesem Bild. Der Code für das folgende Beispiel ist in `configs/apps/gladevcp/animated-backdrop` zu finden:

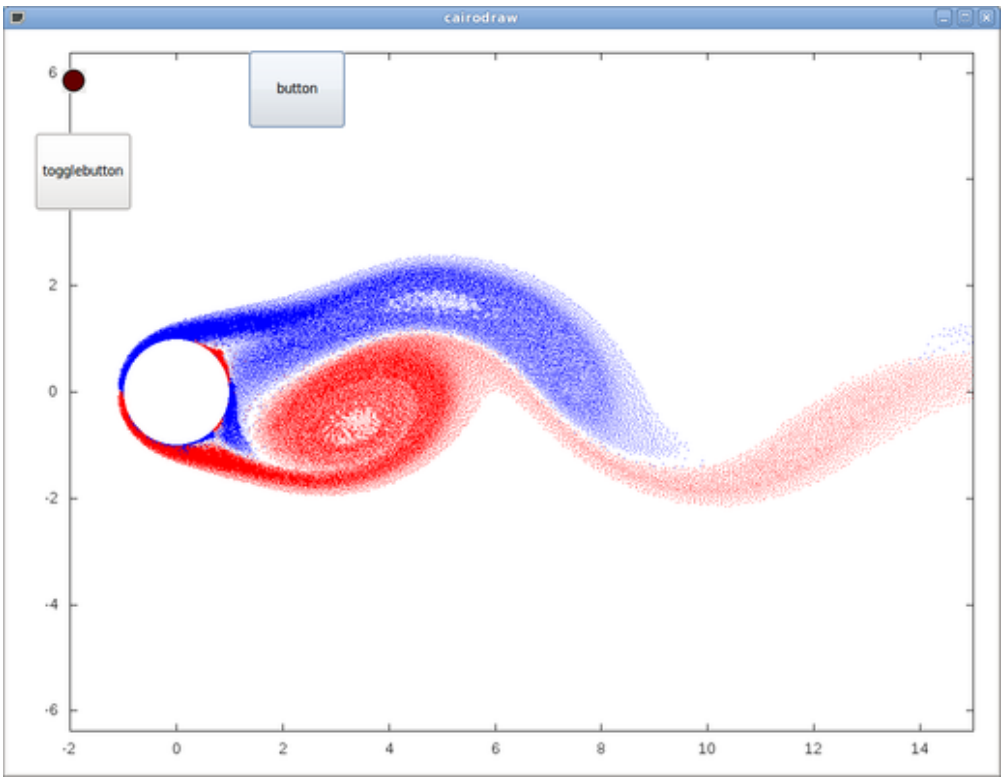


Figure 270. HAL-Widgets in einem Bitmap Beispiel

12.3.7. Referenz zu Aktions-Widgets

GladeVCP enthält eine Sammlung von "vorgefertigten Aktionen" (engl. canned actions) namens **VCP Action Widgets** für den Glade-Benutzeroberflächeneditor.

NOTE	Anders als die mit HAL-Pins interagierenden HAL-Widgets, interagieren VCP-Aktionen mit LinuxCNC und dem G-Code-Interpreter.

VCP Action Widgets sind von dem `Gtk.Action` Widget abgeleitet.

Das Action-Widget in Kürze:

- Es ist ein in Glade verfügbares Objekt
- Es hat für sich genommen kein optisches Erscheinungsbild
- Sein Zweck: Eine sichtbare, empfindliche UI-Komponente wie ein Menü, eine Werkzeugschaltfläche oder eine Schaltfläche mit einem Befehl zu verknüpfen. Siehe die Eigenschaft "General→Related→Action" dieser Widgets.
- Die "vorgefertigte Aktion" (nicht-veränderbares Makro, engl. canned action) wird ausgeführt, wenn die zugehörige UI-Komponente ausgelöst wird (Tastendruck, Menüklick..).
- Es bietet eine einfache Möglichkeit, Befehle auszuführen, ohne auf Python-Programmierung zurückgreifen zu müssen.

Das Erscheinungsbild der VCP-Aktionen in Glade sieht in etwa wie folgt aus:

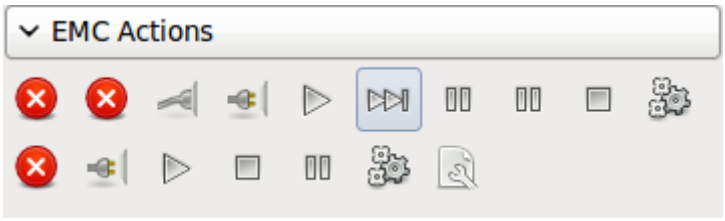


Figure 271. Action-Widgets

Hover über Werkzeugspitzen (engl. tooltips) liefert eine Beschreibung.

VCP Action-Widgets

VCP Action-Widgets sind einmalige Widgets. Sie implementieren eine einzelne Aktion und sind für die Verwendung in einfachen Schaltflächen, Menüeinträgen oder Options-/Kontrollgruppen vorgesehen.

VCP Action Python

Dieses Widget wird verwendet, um kleinen, beliebigen Python-Code auszuführen.

Der Befehlsstring kann spezielle Schlüsselwörter für den Zugriff auf wichtige Funktionen enthalten.

- *ACTION* für den Zugriff auf die ACTION-Befehlsbibliothek.
- `GSTAT` für den Zugriff auf die Gstat-Bibliothek für Statusmeldungen.

- *INFO* für den Zugriff auf gesammelte Daten aus der INI-Datei.
- *HAL* für den Zugriff auf das HAL `linuxcnc` Python-Modul.
- `STAT` für den Zugriff auf LinuxCNC's rohen Status über das LinuxCNC Python Modul.
- `CMD` für den Zugriff auf LinuxCNC-Befehle über das LinuxCNC-Python-Modul.
- `EXT` für den Zugriff auf die Handler-Dateifunktionen, falls verfügbar.
- `linuxcnc` für den Zugriff auf das LinuxCNC-Python-Modul.
- `self` für den Zugriff auf die Widget-Instanz.
- *dir* für den Zugriff auf die Handler-Attributliste.

Es gibt Optionen für

- wählen Sie aus, wann das Widget aktiv sein soll,
- den Modus einstellen, bevor der Befehl ausgeführt wird.

Beispiel für einen Befehl, um einfach eine Nachricht auf dem Terminal zu auszugeben:

```
print('Aktion aktiviert')
```

Beispiel für einen Befehl, um die Maschine in den Aus-Zustand zu versetzen:

```
CMD.state(linuxcnc.STATE_OFF)
```

Beispiel für einen Befehl für den Aufruf einer Handler-Funktion, die Daten übergibt:

```
EXT.on_button_press(self, 100)
```

Sie können ein Semicolon verwenden, um mehrere Befehle zu trennen;

```
print('Set Machine Off');CMD.state(linuxcnc.STATE_OFF)
```

Weitere Informationen zu INFO und ACTION finden Sie hier: GladeVCP Libraries modules.

Weitere Informationen zu GStat finden Sie hier: GStat.

VCP ToggleAction-Widgets

Dies sind **bi-modale** Widgets. Sie implementieren zwei Aktionen oder verwenden einen zweiten (normalerweise `gedrückten`) Zustand, um anzuzeigen, dass gerade eine Aktion ausgeführt wird. Toggle-Aktionen sind für die Verwendung in `ToggleButtons`, `ToggleToolButtons` oder zum Umschalten von Menüpunkten gedacht. Ein einfaches Beispiel ist die `ESTOP` (engl. für Notaus) Umschalttaste.

Derzeit sind die folgenden Widgets verfügbar:

- Der `ESTOP` Toggle sendet `ESTOP` oder `ESTOP_RESET` Befehle an LinuxCNC, abhängig von seinem Zustand.

- Die Umschaltfunktion `ON/OFF` sendet die Befehle `STATE_ON` und `STATE_OFF`.
- `Pause/Fortsetzen` sendet die Befehle `AUTO_PAUSE` oder `AUTO_RESUME`.

Die folgenden Toggle-Aktionen haben nur einen zugehörigen Befehl und verwenden den Zustand "gedrückt", um anzuzeigen, dass der angeforderte Vorgang ausgeführt wird:

- Der `Run`-Toggle sendet einen `AUTO_RUN`-Befehl und wartet im `gedrückten` Zustand, bis der Interpreter wieder im Leerlauf ist.
- Der `Stop`-Schalter ist inaktiv, bis der Interpreter in den aktiven Zustand übergeht (d.h. G-Code ausführt) und dem Benutzer dann erlaubt, den Befehl `AUTO_ABORT` zu senden.
- Der `MDI`-Umschalter sendet einen bestimmten MDI-Befehl und wartet im inaktiven Zustand "gedrückt" auf dessen Ausführung.

Die Action_MDI Toggle und Action_MDI Widgets

Diese Widgets bieten eine Möglichkeit, beliebige MDI-Befehle auszuführen.
Das `Action_MDI`-Widget wartet nicht auf die Beendigung des Befehls, wie es das `Action_MDI`-Toggle tut, das deaktiviert bleibt, bis der Befehl beendet ist.

Ein einfaches Beispiel: Ausführen eines MDI-Befehls bei Button-Druck

`configs/apps/gladevcp/mdi-command-example/whoareyou.ui` ist eine Glade UI-Datei, welche die Grundlagen vermittelt:

1. Öffnen Sie es in Glade und studieren Sie, wie es gemacht wird.
2. Starten Sie AXIS, und starten Sie es dann von einem Terminalfenster aus mit `gladevcp whoareyou.ui`.
3. Sehen Sie sich die Aktion `hal_action_mdi1` und ihre Eigenschaft `MDI command` an - diese führt einfach `(MSG, "Hi, I'm an VCP_Action_MDI")` aus, so dass in AXIS ein Nachrichten-Popup erscheinen sollte, etwa so:

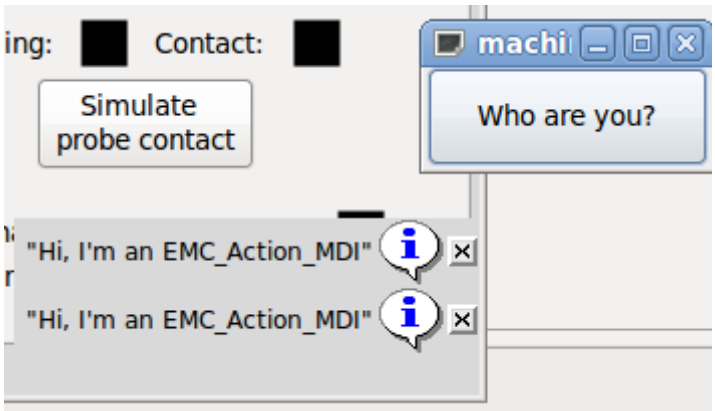


Figure 272. Action_MDI Einfaches Beispiel

Sie werden feststellen, dass die mit der Aktion Action_MDI verbundene Schaltfläche ausgegraut ist, wenn die Maschine ausgeschaltet ist, sich im E-Stop befindet oder der Interpreter läuft. Sie wird automatisch aktiv, wenn die Maschine eingeschaltet ist und sich nicht mehr im Notaus-Modus befindet

und das Programm im Leerlauf ist.

Parameterübergabe mit Action_MDI- und ToggleAction_MDI-Widgets

Optional können bei "MDI Befehl"-Zeichenketten Parameter ersetzt werden, bevor sie an den Interpreter übergeben werden. Parameter können derzeit Namen von HAL-Pins in der GladeVCP-Komponente sein. So funktioniert es:

- Nehmen Sie an, Sie haben eine *HAL SpinBox* mit dem Namen `Geschwindigkeit`, und Sie wollen deren aktuellen Wert als Parameter in einem MDI-Befehl übergeben.
- The HAL SpinBox will have a REAL-type HAL pin named speed-f (see HalWidgets description).
- Um diesen Wert im MDI-Befehl zu ersetzen, fügen Sie den Namen des HAL-Pins in der folgenden Weise ein: ${pin-name}`
- für die obige HAL SpinBox könnten wir `(MSG, "Die Geschwindigkeit ist: ${geschwindigkeit-f}")` verwenden, um zu zeigen, was passiert.

Die Beispiel-UI-Datei ist "configs/apps/gladevcp/mdi-command-example/speed.ui". So sieht das Ergebnis aus, wenn man sie ausführt:

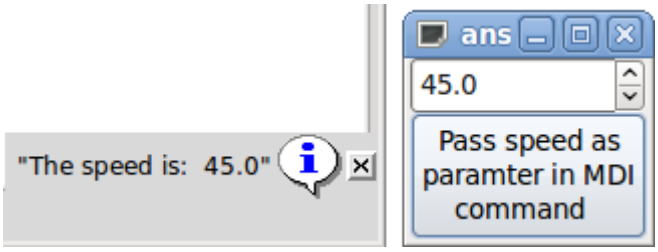


Figure 273. Action_MDI Parameterübergabe Beispiel

Ein fortgeschrittenes Beispiel: Übergabe von Parametern an eine O-Wort-Unterroutine

Es ist völlig in Ordnung, eine O-Wort-Unterroutine in einem MDI-Befehl aufzurufen und HAL-Pin-Werte als aktuelle Parameter zu übergeben. Eine Beispiel-UI-Datei befindet sich in `configs/apps/gladevcp/mdi-command-example/owordsub.ui`.

Legen Sie `nc_files/gladevcp_lib/oword.ngc` so ab, dass AXIS es finden kann, und führen Sie `gladevcp owordsub.ui` in einem Terminalfenster aus. Das sieht dann so aus:

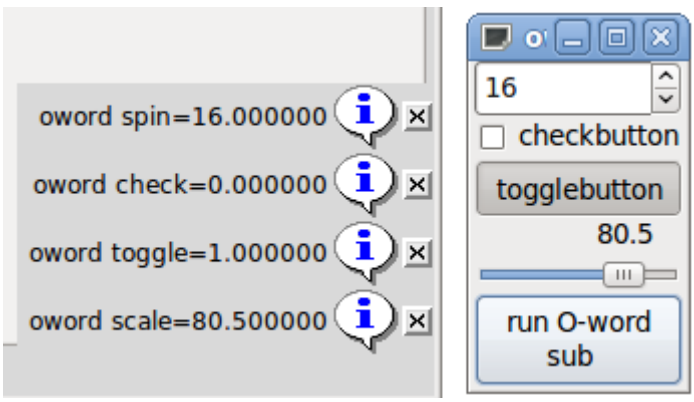


Figure 274. Action_MDI Erweitertes Beispiel

Vorbereitung einer MDI-Aktion und anschließendes Aufräumen

Der LinuxCNC G-Code-Interpreter hat einen einzigen globalen Satz von Variablen, wie Vorschub, Spindeldrehzahl, relative/absolute Modus und andere. Wenn Sie G-Code-Befehle oder O-Wort-Subs verwenden, könnten einige dieser Variablen durch den Befehl oder Unterprogramm geändert werden - zum Beispiel wird ein Antasten Unterprogramm sehr wahrscheinlich den Vorschubwert ziemlich niedrig eingestellt. Ohne weitere Vorkehrungen wird Ihre vorherige Vorschubeinstellung durch den Wert des Sondierungsunterprogramms überschrieben.

Um mit diesem überraschenden und unerwünschten Nebeneffekt eines bestimmten O-Wort-Unterprogramms oder einer G-Code-Anweisung, die mit einem LinuxCNC ToggleAction_MDI ausgeführt wird, umzugehen, können Sie pre-MDI- und post-MDI-Handler mit einem bestimmten LinuxCNC ToggleAction_MDI verbinden. Diese Handler sind optional und bieten die Möglichkeit, den Zustand vor der Ausführung der MDI-Aktion zu speichern und danach wieder auf die vorherigen Werte zurückzusetzen. Die Signalnamen sind `mdi-command-start` und `mdi-command-stop`; die Namen der Handler können in Glade wie jeder andere Handler gesetzt werden.

Hier ist ein Beispiel, wie ein Vorschubwert durch solche Handler gespeichert und wiederhergestellt werden könnte (beachten Sie, dass LinuxCNC Befehls- und Statuskanäle als `self.linuxcnc` und `self.stat` durch die VCP_ActionBase Klasse verfügbar sind):

```
    def on_mdi_command_start(self, action, userdata=None):
        action.stat.poll()
        self.start_feed = action.stat.settings[1]

    def on_mdi_command_stop(self, action, userdata=None):
        action.linuxcnc.mdi('F%.1f' % (self.start_feed))
        while action.linuxcnc.wait_complete() == -1:
            pass
```

Nur das Toggle-Widget `Action_MDI` unterstützt diese Signale.

NOTE In einer späteren Version von LinuxCNC, werden die neuen M-Codes M70-M72 verfügbar sein. Sie machen das Speichern von Zustand vor einem Unterprogramm aufrufen, und Wiederherstellen von Zustand bei der Rückkehr viel einfacher.

Verwendung des LinuxCNC Stat-Objekts zum Umgang mit Statusänderungen

Viele Aktionen hängen vom LinuxCNC-Status ab - ist es im manuellen, MDI- oder Auto-Modus? Läuft ein Programm, pausiert es oder ist es im Leerlauf? Sie können keinen MDI-Befehl starten, während ein G-Code-Programm läuft, also muss dies beachtet werden. Viele LinuxCNC-Aktionen kümmern sich selbst darum, und die zugehörigen Schaltflächen und Menüeinträge sind deaktiviert, wenn die Operation gerade nicht möglich ist.

Bei der Verwendung von Python-Ereignishandlern - die sich auf einer niedrigeren Ebene als Actions befinden - muss man sich selbst um den Umgang mit Statusabhängigkeiten kümmern. Für diesen Zweck gibt es das LinuxCNC Stat-Widget: um LinuxCNC-Statusänderungen mit Event-Handlern zu verknüpfen.

LinuxCNC Stat hat keine sichtbare Komponente - Sie fügen es einfach mit Glade zu Ihrer

Benutzeroberfläche hinzu. Einmal hinzugefügt, können Sie Handler mit den folgenden Signalen verknüpfen:

- zustandsbezogen:
  - `state-estop`: ausgegeben, wenn die Notaus-Bedingung eintritt,
  - `state-estop-reset`: ausgegeben, wenn die Maschine zurückgesetzt wird,
  - "state-on": wird beim Einschalten des Geräts ausgegeben,
  - `state-off`: wird ausgegeben, wenn die Maschine ausgeschaltet wird.
- Modus-bezogen:
  - *mode-manual*: wird ausgegeben, wenn LinuxCNC in den manuellen Modus wechselt,
  - `mode-mdi`: ausgegeben, wenn LinuxCNC in den MDI-Modus wechselt,
  - `mode-auto`: ausgegeben, wenn LinuxCNC in den automatischen Modus wechselt,
- Interpreter-bezogen: wird ausgegeben, wenn der G-Code-Interpreter in diesen Modus wechselt
  - `interp-run`
  - `interp-idle`
  - `interp-paused`
  - `interp-reading`
  - `interp-waiting`
  - `file-loaded`
  - `line-changed`
- Referenzfart-bezogen: ausgegeben, wenn LinuxCNC referenziert ist oder nicht
  - `all-homed`
  - `nicht-all-homed`

12.3.8. GladeVCP-Programmierung

Benutzerdefinierte Aktionen

Die meisten Widgetsets und die dazugehörigen Editoren für die Benutzeroberfläche unterstützen das Konzept der Callbacks, d.h. Funktionen im vom Benutzer geschriebenen Code, die ausgeführt werden, wenn in der Benutzeroberfläche "etwas passiert" - Ereignisse wie Mausklicks, eingegebene Zeichen, Mausbewegungen, Timer-Ereignisse, das Ein- und Ausblenden von Fenstern und so weiter.

HAL-Ausgabe-Widgets bilden typischerweise Eingabe-Ereignisse wie einen Tastendruck mittels eines solchen - vordefinierten - Callbacks auf eine Wertänderung des zugehörigen HAL-Pins ab. In PyVCP ist dies wirklich die einzige Art der Ereignisbehandlung, die unterstützt wird - etwas Komplexeres, wie die Ausführung von MDI-Befehlen zum Aufruf eines G-Code-Unterprogramms, wird nicht unterstützt.

Innerhalb von GladeVCP sind HAL-Pin-Änderungen nur ein Typ der allgemeinen Klasse von Ereignissen (Signale genannt) in GTK+. Die meisten Widgets können solche Signale auslösen, und der Glade-Editor

unterstützt die Verknüpfung eines solchen Signals mit einem Python-Methoden- oder Funktionsnamen.

Wenn Sie sich entscheiden, benutzerdefinierte Aktionen zu verwenden, ist es Ihre Aufgabe, ein Python-Modul zu schreiben, dessen Klassenmethoden - oder im einfachen Fall nur Funktionen - in Glade als Event-Handler referenziert werden können. GladeVCP bietet eine Möglichkeit, Ihr(e) Modul(e) beim Start zu importieren und wird Ihre Event-Handler automatisch mit den Widgetsignalen verknüpfen, wie sie in der Glade-UI-Beschreibung festgelegt sind.

Core-Bibliothek

Es gibt drei Bibliotheken mit Funktionen, die zur Programmierung von GladeVCP verwendet werden können.

- *Info*: sammelt Details aus der INI-Datei.
- *Action*: Eine Sammlung von Funktionen zum Ändern von LinuxCNC-Zuständen.
- *Status*: Meldet den Status von LinuxCNC. Es führt intern "Gstat" aus ("wrapt").

Importieren und Instanziieren der Bibliotheken:

```
from gladevcp.core import Info, Action

ACTION = Action()
INFO = Info()
```

Verwendung der Bibliotheksfunktionen:

```
print(INFO.MACHINE_IS_METRIC)
ACTION.SET_ERROR_MESSAGE('Something went wrong')
```

Weitere Informationen finden Sie hier : GladeVCP Libraries modules. Es gibt eine Beispielkonfiguration, welche die Verwendung der Kernbibliothek mit GladeVCPs Action-Python-Widgets und mit einer Python-Handler-Datei demonstriert. Versuchen Sie, *sim/axis/gladevcp/gladevcp_panel_tester* zu laden.

Ein Beispiel: Hinzufügen benutzerdefinierter Callback-Funktionen in Python

Dies ist nur ein minimales Beispiel, um die Idee zu vermitteln - Details werden im restlichen Teil dieses Abschnitts erläutert.

GladeVCP kann nicht nur HAL-Pins manipulieren oder anzeigen, man kann auch reguläre Event-Handler in Python schreiben. Dies kann unter anderem zur Ausführung von MDI-Befehlen verwendet werden. So wird es gemacht:

Schreiben Sie ein Python-Modul wie folgt und speichern Sie es z. B. als handlers.py:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
```

```
    gtkobj.set_label("hits: %d" % nhits)
```

Definieren Sie in Glade eine Schaltfläche oder eine HAL-Schaltfläche, wählen Sie die Registerkarte "Signale" und wählen Sie in den Eigenschaften von GtkButton die Zeile "pressed". Geben Sie dort "on_button_press" ein und speichern Sie die Glade-Datei.

Fügen Sie dann die Option *-u handlers.py* in die GladeVCP-Befehlszeile ein. Wenn Ihre Event-Handler über mehrere Dateien verteilt sind, fügen Sie einfach mehrere *-u <pyfilename>* Optionen hinzu.

Wenn Sie nun auf die Schaltfläche drücken, sollte sich ihre Beschriftung ändern, da sie in der Callback-Funktion festgelegt wurde.

Was das Flag `-u` bewirkt: alle Python-Funktionen in dieser Datei werden gesammelt und als potentielle Callback-Handler für Ihre Gtk-Widgets eingerichtet - sie können über die Glade-Registerkarten "Signale" referenziert werden. Die Callback-Handler werden mit der jeweiligen Objektinstanz als Parameter aufgerufen, wie die GtkButton-Instanz oben, so dass Sie jede GtkButton-Methode von dort aus anwenden können.

Oder machen Sie etwas Nützlicheres, wie den Aufruf eines MDI-Befehls!

HAL-Wertänderungs-Ereignisse

HAL-Eingangs-Widgets, wie z.B. eine LED, assoziieren automatisch ihren HAL-Pin-Status (an/aus) mit dem optischen Erscheinungsbild des Widgets (LED leuchtet/dunkelt).

Über diese eingebaute Funktionalität hinaus kann man jedem HAL-Pin, auch denen von vordefinierten HAL-Widgets, einen Änderungs-Callback zuordnen. Dies passt gut zu der ereignisgesteuerten Struktur einer typischen Widget-Anwendung: Jede Aktivität, sei es ein Mausklick, eine Taste, ein abgelaufener Timer oder die Änderung des Wertes eines HAL-Pins, erzeugt einen Callback und wird durch denselben orthogonalen Mechanismus behandelt.

Für benutzerdefinierte HAL-Pins, die nicht mit einem bestimmten HAL-Widget verbunden sind, lautet der Signalname *value-changed*. Siehe den Abschnitt AL Pins hinzufügen weiter unten für Details.

HAL Widgets werden mit einem vordefinierten Signal namens *hal-pin-changed* geliefert. Siehe den Abschnitt HAL Widgets für Details.

Programmiermodell

Das Gesamtkonzept sieht folgendermaßen aus:

- Entwerfen Sie Ihre Benutzeroberfläche mit Glade, und legen Sie Signal-Handler fest, wenn Sie Aktionen mit einem Widget verbinden möchten.
- Schreiben Sie ein Python-Modul, das aufrufbare Objekte enthält (siehe "Handler-Modelle" unten).
- Übergeben Sie den Pfadnamen Ihres Moduls an GladeVCP mit der Option *-u <modul>*.
- GladeVCP importiert das Modul, prüft es auf Signalhandler und verbindet sie mit dem Widgetbaum.
- Die Hauptereignisschleife wird ausgeführt.

Das einfache Handler-Modell

Für einfache Aufgaben reicht es aus, Funktionen zu definieren, die nach den Glade-Signalhandlern benannt sind. Diese werden aufgerufen, wenn das entsprechende Ereignis im Widgetbaum eintritt. Hier ist ein triviales Beispiel - es nimmt an, dass das *pressed* Signal eines Gtk Buttons oder HAL Buttons mit einem Callback namens *on_button_press* verknüpft ist:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
    gtkobj.set_label("hits: %d" % nhits)
```

Fügen Sie diese Funktion in eine Python-Datei ein und führen Sie sie wie folgt aus:

```
gladevcp -u <myhandler>.py mygui.ui
```

Beachten Sie, dass die Kommunikation zwischen Handlern über globale Variablen erfolgen muss, was nicht gut skalierbar und absolut unpythonisch ist. Aus diesem Grund haben wir das klassenbasierte Handler-Modell entwickelt.

Das klassenbasierte Handler-Modell

Die Idee dabei ist: Handler werden mit Klassenmethoden verknüpft. Die zugrundeliegende(n) Klasse(n) werden beim Start von GladeVCP instanziiert und inspiziert und als Signal-Handler mit dem Widget-Baum verknüpft. Die Aufgabe ist nun also zu schreiben:

- Eine oder mehrere Klassendefinition(en) mit einer oder mehreren Methoden, in einem Modul oder aufgeteilt auf mehrere Module,
- eine Funktion *get_handlers* in jedem Modul, die eine Liste von Klasseninstanzen an GladeVCP zurückgibt - ihre Methodennamen werden mit Signalhandlern verknüpft.

Hier ist ein minimales benutzerdefiniertes Handler-Beispielmodul:

```
class MyCallbacks :
    def on_this_signal(self,obj,data=None):
        print("this_signal happened, obj=",obj)

def get_handlers(halcomp,builder,useropts):
    return [MyCallbacks ()]
```

Jetzt wird *on_this_signal* als Signalhandler für Ihren Widgetbaum verfügbar sein.

GladeVCP-spezifische Signale

Für GladeVCP-Panels, die auf HAL-Eingaben reagieren, kann es wichtig sein, dass der Handler-Code erkennen kann, dass das GladeVCP-Panel gerade aktiv und angezeigt ist. Zum Beispiel könnte ein Panel innerhalb der Touchy-Schnittstelle durchaus eine Aktion ausführen müssen, wenn der mit touchy.cycle-start verbundene Schalter betätigt wird (so wie die nativen Tabs unterschiedlich auf die gleiche

Schaltfläche reagieren).
Um dies zu ermöglichen, wird ein Signal von der GUI (zum Zeitpunkt des Schreibens nur Touchy) an die eingebettete Registerkarte gesendet. Das Signal ist vom Typ "Gladevcp" und die beiden gesendeten Nachrichten sind "Sichtbar" und "Ausgeblendet". (Beachten Sie, dass die Signale eine feste Länge von 20 Zeichen haben, so dass nur die ersten Zeichen in einem Vergleich verwendet werden sollten, daher die [:7] unten). Ein Beispiel für einen Handler für diese Signale ist:

```
    # This catches our messages from another program
    def event(self,w,event):
        print(event.message_type,event.data)
        if event.message_type == 'Gladevcp':
            if event.data[:7] == 'Visible':
                self.active = True
            else:
                self.active = False

    # connect to client-events from the host GUI
    def on_map_event(self, widget, data=None):
        top = widget.get_toplevel()
        print("map event")
        top.connect('client-event', self.event)
```

Das get_handlers Protokoll

Wenn GladeVCP bei der Modulinspektion eine Funktion `get_handlers` findet, ruft es diese wie folgt auf:

```
get_handlers(halcomp,builder,useropts)
```

Die Argumente sind:

- `halcomp` - bezieht sich auf die im Bau befindliche HAL-Komponente,
- `builder` - Widget-Baum - Ergebnis des Lesens der UI-Definition (entweder auf ein Objekt vom Typ GtkBuilder oder libglade verweisend),
- `useropts` - eine Liste von Zeichenfolgen, die von der GladeVCP-Befehlszeile mit der Option *-U <useropts>* gesammelt werden.

GladeVCP untersucht dann die Liste der Klasseninstanzen und ruft deren Methodennamen ab. Qualifizierende Methodennamen werden als Signalhandler mit dem Widgetbaum verbunden. Nur Methodennamen, die nicht mit einem _ (Unterstrich) beginnen, werden berücksichtigt.

Beachten Sie, dass unabhängig davon, ob Sie das libglade- oder das neue GtkBuilder-Format für Ihre Glade-Benutzeroberfläche verwenden, Widgets immer als `builder.get_object(<widgetname>)` bezeichnet werden können. Außerdem ist die komplette Liste der Widgets als `builder.get_objects()` verfügbar, unabhängig vom UI-Format.

Initialisierungssequenz

Es ist wichtig zu wissen, in welchem Zustand die Funktion `get_handlers()` aufgerufen wird, damit Sie wissen, was Sie dort sicher tun können und was nicht. Zunächst werden die Module in der Befehlszeilenreihenfolge importiert und initialisiert. Nach erfolgreichem Import wird `get_handlers()` im folgenden Zustand aufgerufen:

- Der Widgetbaum ist erstellt, aber noch nicht realisiert (es wurde noch kein Toplevel `window.show()` ausgeführt).
- Die halcomp HAL-Komponente ist eingerichtet und alle Pins des HAL-Widgets wurden bereits hinzugefügt.
- Es ist sicher, weitere HAL-Pins hinzuzufügen, da *halcomp.ready()* zu diesem Zeitpunkt noch nicht aufgerufen wurde, so dass Sie Ihre eigenen Pins hinzufügen können, zum Beispiel in der Klasse *init()* -Methode.

Nachdem alle Module importiert und die Methodennamen extrahiert wurden, werden die folgenden Schritte durchgeführt:

- Alle qualifizierenden Methodennamen werden mit *connect_signals()/signal_autoconnect()* mit dem Widget-Baum verbunden (abhängig von der Art der importierten Benutzeroberfläche - GtkBuilder im Vergleich zum alten libglade-Format).
- Die HAL-Komponente wird mit halcomp.ready() abgeschlossen.
- Wenn eine Fenster-ID als Argument übergeben wurde, wird der Widget-Baum neu geparented, um in diesem Fenster zu laufen, und Glades Toplevel window1 wird aufgegeben (siehe FAQ).
- Wenn eine HAL-Befehlsdatei mit `-H halfile` übergeben wurde, wird sie mit halcmd ausgeführt.
- Die Gtk-Hauptschleife wird ausgeführt.

Wenn also Ihre Handler-Klasse initialisiert wird, sind alle Widgets vorhanden, aber noch nicht realisiert (auf dem Bildschirm angezeigt). Und die HAL-Komponente ist auch noch nicht fertig, so dass es unsicher ist, auf die Werte der Pins in Ihrer Methode "*init*_()" zuzugreifen.

Wenn Sie einen Callback haben wollen, der beim Programmstart ausgeführt wird, nachdem es sicher ist, auf die HAL-Pins zuzugreifen, dann verbinden Sie einen Handler mit dem realize-Signal des Top-Level-Fensters1 (was sein einziger wirklicher Zweck sein könnte). An diesem Punkt ist GladeVCP mit allen Setup-Aufgaben fertig, die HAL-Datei wurde ausgeführt, und GladeVCP ist dabei, in die Gtk-Hauptschleife einzutreten.

Mehrere Callbacks mit demselben Namen

Innerhalb einer Klasse müssen die Methodennamen eindeutig sein. Es ist jedoch in Ordnung, wenn mehrere Klasseninstanzen mit identisch benannten Methoden durch get_handlers() an GladeVCP übergeben werden. Wenn das entsprechende Signal auftritt, werden diese Methoden in der Definitionsreihenfolge aufgerufen - Modul für Modul, und innerhalb eines Moduls in der Reihenfolge, in der die Klasseninstanzen von "get_handlers()" zurückgegeben werden.

Die GladeVCP `-U <useropts>` Flag

Anstatt GladeVCP für jede denkbare Option zu erweitern, die für eine Handler-Klasse potentiell nützlich sein könnte, können Sie das Flag -U <Benutzeroption> verwenden (auf Wunsch wiederholt). Dieses Flag sammelt eine Liste von <useroption>-Strings. Diese Liste wird an die Funktion get_handlers() übergeben (Argument useropts). Es steht Ihrem Code frei, diese Zeichenfolgen nach eigenem Ermessen zu interpretieren. Eine mögliche Verwendung wäre, sie in der Funktion `get_handlers()` wie folgt an die Python-Funktion exec zu übergeben:

```
debug = 0
...
def get_handlers(halcomp,builder,useropts):
    ...
    global debug # assuming there's a global var
    for cmd in useropts:
        exec cmd in globals()
```

Auf diese Weise können Sie beliebige Python-Anweisungen an Ihr Modul übergeben, zum Beispiel durch die Option `gladevcp -U`:

```
gladevcp -U debug=42 -U "print 'debug=%d' % debug" ...
```

Dies sollte debug auf 2 setzen und bestätigen, dass Ihr Modul es tatsächlich getan hat.

Persistente Variablen in GladeVCP

Ein ärgerlicher Aspekt von GladeVCP in seiner früheren Form und PyVCP ist die Tatsache, dass Sie Werte und HAL-Pins durch Texteingabe, Schieberegler, Spin-Boxen, Toggle-Buttons usw. ändern können, aber ihre Einstellungen werden nicht gespeichert und beim nächsten Lauf von LinuxCNC wiederhergestellt - sie beginnen mit dem Standardwert, wie in der Panel-oder Widget-Definition eingestellt.

GladeVCP verfügt über einen einfach zu bedienenden Mechanismus zum Speichern und Wiederherstellen des Zustands von HAL-Widgets und Programmvariablen (in der Tat jedes Instanzattribut vom Typ int, float, bool oder string).

Dieser Mechanismus verwendet das weit verbreitete INI-Dateiformat, um dauerhafte Attribute zu speichern und wieder zu laden.

Persistenz, Programmversionen und die Signaturprüfung

Stellen Sie sich vor, Sie benennen Widgets in Glade um, fügen sie hinzu oder löschen sie: Eine INI-Datei aus einer früheren Programmversion oder eine völlig andere Benutzeroberfläche könnte den Zustand nicht richtig wiederherstellen, da sich Variablen und Typen geändert haben könnten.

GladeVCP erkennt diese Situation durch eine Signatur, die von allen Objektnamen und -typen abhängt, die gespeichert sind und wiederhergestellt werden sollen. Im Falle einer Nichtübereinstimmung der Signatur wird eine neue INI-Datei mit Standardeinstellungen erzeugt.

Verwendung persistenter Variablen

Wenn Sie möchten, dass der Status des Gtk-Widgets, die Werte des HAL-Widgets-Ausgabepins und/oder die Klassenattribute Ihrer Handler-Klasse über Aufrufe hinweg erhalten bleiben, gehen Sie wie folgt vor:

- Importieren Sie das Modul *gladevcp.persistence*.
- Entscheiden Sie, welche Instanzattribute und deren Standardwerte Sie beibehalten wollen, falls vorhanden.
- Entscheiden Sie, welche Widgets ihren Zustand beibehalten sollen.
- Beschreiben Sie diese Entscheidungen in der `__init__()` Methode Ihrer Handler-Klasse durch ein verschachteltes Wörterbuch wie folgt:

```
def __init__(self, halcomp,builder,useropts):
    self.halcomp = halcomp
    self.builder = builder
    self.useropts = useropts
    self.defaults = {
        # die folgenden Namen werden als Methodenattribute gespeichert/wiederhergestellt
        # Der Mechanismus zum Speichern/Wiederherstellen ist stark typisiert - der Typ
der Variablen wird vom Typ des
        # Initialisierungswertes abgeleitet. Derzeit unterstützte Typen sind: int, float,
bool, string
        IniFile.vars : { 'nhits' : 0, 'a': 1.67, 'd': True ,'c' : "ein String"},
        # zum Speichern/Wiederherstellen aller Widgets, die auch nur im Entferntesten
Sinn machen könnten, fügen Sie dies hinzu:
        IniFile.widgets : widget_defaults(builder.get_objects())
        # Eine sinnvolle Alternative wäre es, nur den Zustand aller HAL-Ausgabe-Widgets
beizubehalten:
        # IniFile.widgets: widget_defaults(select_widgets(self.builder.get_objects(),
hal_only=True,output_only = True)),
    }
```

Dann verknüpfen Sie eine INI-Datei mit diesem Deskriptor:

```
self.ini_filename = __name__ + '.ini'
self.ini = IniFile(self.ini_filename,self.defaults,self.builder)
self.ini.restore_state(self)
```

Nach `restore_state()` werden die Attribute von self gesetzt, wenn sie wie folgt ablaufen:

```
self.nhits = 0
self.a = 1.67
self.d = True
self.c = "eine Zeichenkette"
```

Beachten Sie, dass die Typen gespeichert und bei der Wiederherstellung beibehalten werden. In diesem Beispiel wird davon ausgegangen, dass die INI-Datei nicht vorhanden war oder die Standardwerte aus self.defaults enthielt.

Nach dieser Beschwörung können Sie die folgenden IniFile-Methoden verwenden:

ini.save_state(obj)

Speichert die Attribute von objs gemäß dem `IniFile.vars`-Wörterbuch und den Zustand des Widgets wie in `IniFile.widgets` beschrieben in `self.defaults`.

ini.create_default_ini()

Erstellen Sie eine INI-Datei mit Standardwerten.

ini.restore_state(obj)

HAL out Pins und obj's Attribute wie oben gespeichert/initialisiert auf Standard zurücksetzen.

Speichern des Status beim Herunterfahren von GladeVCP

Um den Zustand des Widgets und/oder der Variablen beim Beenden zu speichern, gehen Sie wie folgt vor:

- Wählen Sie ein Interieur-Widget aus (Typ ist nicht wichtig, z. B. eine Tabelle).
- Wählen Sie auf der Registerkarte "Signale" die Option "GtkObject". In der ersten Spalte sollte ein *destroy*-Signal angezeigt werden.
- Fügen Sie den Namen des Handlers, z. B. *on_destroy*, in die zweite Spalte ein.
- Fügen Sie einen Python-Handler wie unten beschrieben hinzu:

```
import gtk
...
def on_destroy(self,obj,data=None):
    self.ini.save_state(self)
```

Dadurch wird der Status gespeichert und GladeVCP ordnungsgemäß heruntergefahren, unabhängig davon, ob das Panel in AXIS eingebettet oder ein eigenständiges Fenster ist.

CAUTION Verwenden Sie nicht `window1` (das Toplevel-Fenster), um ein`destroy` (engl. für zerstören)-Event zu verbinden. Aufgrund der Art und Weise, wie ein GladeVCP-Panel mit AXIS interagiert, wenn ein Panel in AXIS eingebettet ist, wird **window1 destroy-Ereignisse nicht richtig empfangen**. Da jedoch beim Herunterfahren alle Widgets zerstört werden, reicht jedes beliebige aus. Empfohlen: Verwenden Sie ein Widget der zweiten Ebene - zum Beispiel, wenn Sie einen Tabellencontainer in Ihrem Panel haben, verwenden Sie diesen.

Wenn Sie die GladeVCP-Anwendung das nächste Mal starten, sollten die Widgets in dem Zustand angezeigt werden, in dem sie beim Schließen der Anwendung waren.

CAUTION Die *GtkWidget*-Zeile hat ein ähnlich klingendes *destroy-event* - **nicht verwenden, um sich mit dem *on_destroy*-Handler zu verbinden, es wird nicht funktionieren** - stellen Sie sicher, dass Sie das *destroy*-Ereignis aus der *GtkObject*-Zeile verwenden.

Status speichern, wenn Strg-C gedrückt wird

Standardmäßig reagiert GladeVCP auf ein Ctrl-C-Ereignis mit einem einfachen Beenden - *ohne* den Status zu speichern. Um sicherzustellen, dass dieser Fall abgedeckt ist, fügen Sie einen Handler-Aufruf `on_unix_signal` hinzu, der automatisch bei Ctrl-C (eigentlich bei den Signalen SIGINT und SIGTERM) aufgerufen wird. Beispiel:

```
def on_unix_signal(self,signum,stack_frame):
    print("on_unix_signal(): signal %d received, saving state" % (signum))
    self.ini.save_state(self)
```

Manuelle Bearbeitung von INI-Dateien (.ini)

Sie können dies tun, aber beachten Sie, dass die Werte in self.defaults Ihre Änderungen überschreiben, wenn ein Syntax- oder Typfehler in Ihrer Bearbeitung auftritt. Der Fehler wird erkannt, eine Konsolenmeldung weist auf den Fehler hin und die fehlerhafte INI-Datei wird umbenannt und erhält die Endung .BAD. Nachfolgende fehlerhafte INI-Dateien überschreiben frühere .BAD-Dateien.

Hinzufügen von HAL-Pins

Wenn Sie HAL-Pins benötigen, die nicht mit einem bestimmten HAL-Widget verbunden sind, fügen Sie sie wie folgt hinzu:

```
import hal_glib
...
# in your handler class __init__():
self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.Type.BOOL,
hal.Dir.IN))
```

Um einen Callback zu erhalten, wenn sich der Wert dieses Pins ändert, verknüpfen Sie einen `value-change` (engl. für Wert-Änderung) Callback mit diesem Pin, fügen Sie hinzu:

```
self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

und definieren Sie eine Fallback-Methode (oder -Funktion, in diesem Fall lassen Sie den Parameter `self` weg):

```
# Hinweis: '_' - diese Methode ist für den Widget-Baum nicht sichtbar.
def _on_example_trigger_change(self,pin,userdata=None):
    print("Pin-Wert geändert in:" % (pin.get()))
```

Hinzufügen von Timern

Da GladeVCP Gtk-Widgets verwendet, die sich auf die Basisklasse PyGObject stützen, ist die volle GLib-Funktionalität verfügbar. Hier ist ein Beispiel für einen Timer-Callback:

```
def _on_timer_tick(self,userdata=None):
    ...
```

```
    return True # um den Timer neu zu starten; return False für on-shot
...
# Demonstration eines langsamen Hintergrund-Timers - Granularität ist eine Sekunde
# für einen schnelleren Timer (Granularität 100 ms), verwenden Sie dies:
# GLib.timeout_add(100, self._on_timer_tick,userdata) # 10Hz
GLib.timeout_add_seconds(1, self._on_timer_tick)
```

HAL-Widget-Eigenschaften programmatisch einstellen

Bei Glade werden die Widget-Eigenschaften normalerweise während der Bearbeitung fest eingestellt. Sie können jedoch Widgeteigenschaften zur Laufzeit festlegen, z.B. anhand von INI-Dateiwerten, was normalerweise im Initialisierungscode des Handlers geschieht. Das Setzen von Eigenschaften aus HAL-Pin-Werten ist ebenfalls möglich.

Im folgenden Beispiel (unter der Annahme eines HAL Meter-Widgets mit dem Namen `meter`) wird der Minimalwert des Zählers beim Start über einen INI-Dateiparameter und der Maximalwert über einen HAL-Pin eingestellt, wodurch die Skala des Widgets dynamisch angepasst wird:

```
import linuxcnc
import os
import hal
import hal_glib

class HandlerClass:

    def _on_max_value_change(self,hal_pin,data=None):
        self.meter.max = float(hal_pin.get())
        self.meter.queue_draw() # force a widget redraw

    def __init__(self, halcomp,builder,useropts):
        self.builder = builder

        # HAL pin with change callback.
        # When the pin's value changes the callback is executed.
        self.max_value = hal_glib.GPin(halcomp.newpin('max-value',  hal.Type.REAL, hal
.Dir.IN))
        self.max_value.connect('value-changed', self._on_max_value_change)

        inifile = linuxcnc.ini(os.getenv("INI_FILE_NAME"))
        mmin = inifile.getreal("METER", "MIN", fallback=0.0)
        self.meter = self.builder.get_object('meter')
        self.meter.min = mmin


def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,builder,useropts)]
```

"Wert hat sich geändert"-Callback Funktion mit hal_glib

GladeVCP nutzt die hal_glib-Bibliothek, die dazu verwendet werden kann, ein "watcher" Signal an einen HAL-Eingangspin anzuschließen.
Dieses Signal kann verwendet werden, um eine Funktion zu registrieren, die aufgerufen wird, wenn

sich der Zustand des HAL-Pins ändert.

Man muss das `hal_glib` und das `hal`-Modul importieren:

```
import hal_glib
import hal
```

Erstellen Sie dann einen Pin und verbinden Sie ein *value-changed* Signal (den Watcher) mit einem Funktionsaufruf:

```
class HandlerClass:
    def __init__(self, halcomp,builder,useropts):
        self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.Type
.BOOL, hal.Dir.IN))
        self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

Und eine Funktion haben, die aufgerufen werden soll:

```
    def _on_example_trigger_change(self,pin,userdata=None):
        print("pin value changed to: {}".format(pin.get()))
        print("pin name= {}".format(pin.get_name()))
        print("pin type= {}".format(pin.get_type()))

        # dies kann außerhalb der Funktion aufgerufen werden
        self.example_trigger.get()
```

Beispiele und die Entwicklung Ihrer eigenen GladeVCP-Anwendung

Besuchen Sie `linuxcnc_root_directory/configs/apps/gladevcp` für laufende Beispiele und Startprogramme für Ihre eigenen Projekte.

12.3.9. FAQ

1. *Ich erhalte ein unerwartetes Unmap-Ereignis in meiner Handler-Funktion direkt nach dem Start. Was ist das?*

   Dies ist eine Folge davon, dass in Ihrer Glade UI-Datei die Eigenschaft window1 Visible auf True gesetzt ist und das GladeVCP-Fenster in AXIS oder touchy neu geparentet wird. Der GladeVCP-Widget-Baum wird erstellt, einschließlich eines Fensters der obersten Ebene, und dann in AXIS "reparented", so dass das Fenster der obersten Ebene verwaist herumliegt. Um zu vermeiden, dass dieses nutzlose leere Fenster herumhängt, wird es entmappt (unsichtbar gemacht), was die Ursache für das Unmap-Signal ist, das Sie erhalten. Vorgeschlagene Lösung: window1.visible auf False setzen und ein anfängliches Unmap-Ereignis ignorieren.

2. *Mein GladeVCP-Programm startet, aber es erscheint kein Fenster dort, wo ich es erwarte?*

   Das Fenster, das AXIS für GladeVCP zuweist, erhält die "natürliche Größe" aller seiner untergeordneten Widgets zusammen. Es ist die Aufgabe der untergeordneten Widgets, eine Größe (Breite und/oder Höhe) anzufordern. Allerdings fordern nicht alle Widgets eine Breite größer als 0 an, zum Beispiel das Graph-Widget in seiner aktuellen Form. Wenn es ein solches Widget in Ihrer Glade-Datei gibt und es dasjenige ist, welches das Layout definiert, sollten Sie seine Breite explizit

festlegen. Beachten Sie, dass das Festlegen der Eigenschaften Breite und Höhe von window1 in Glade nicht sinnvoll ist, da dieses Fenster beim Re-Parenting verwaist wird und seine Geometrie daher keine Auswirkungen auf das Layout hat (siehe oben). Generell gilt: Wenn Sie eine UI-Datei manuell mit *gladevcp <uifile>* ausführen und ihr Fenster eine vernünftige Geometrie hat, sollte es auch in AXIS korrekt angezeigt werden.

3. *Ich möchte eine blinkende LED, aber sie blinkt nicht*

   Ich habe das Kontrollkästchen aktiviert, damit die Anzeige im Abstand von 100 ms blinkt. Sie blinkt nicht, und ich erhalte eine Startwarnung: Warning: value "0" of type ‘gint’ is invalid or out of range for property ‘led-blink-rate’ of type ‘gint’? Dies scheint ein Glade-Bug zu sein. Überschreiben Sie einfach das Feld für die Blinkrate und speichern Sie erneut - das funktioniert bei mir.

4. *Mein GladeVCP-Panel in AXIS speichert den Status nicht, wenn ich AXIS schließe, obwohl ich einen on_destroy-Handler definiert habe, der mit dem Fensterzerstörungssignal verbunden ist*

   Sehr wahrscheinlich ist dieser Handler mit window1 verknüpft, das aufgrund des Reparenting für diesen Zweck nicht verwendbar ist. Bitte verknüpfen Sie den on_destroy-Handler mit dem destroy-Signal eines inneren Fensters. Ich habe z.B. ein Notizbuch innerhalb von window1 und habe on_destroy mit dem Zerstörungssignal des Notizbuchs verknüpft, und das funktioniert gut. Für window1 funktioniert es nicht.

5. *Ich möchte die Hintergrundfarbe oder den Text eines HAL_Label-Widgets abhängig von seinem HAL-Pin-Wert einstellen*

   Siehe das Beispiel in configs/apps/gladevcp/colored-label. Das Einstellen der Hintergrundfarbe eines GtkLabel-Widgets (und HAL_Label ist von GtkLabel abgeleitet) ist ein wenig knifflig. Das GtkLabel-Widget hat aus Leistungsgründen kein eigenes Fensterobjekt, und nur Fensterobjekte können eine Hintergrundfarbe haben. Die Lösung ist, das Label in einen EventBox-Container einzuschließen, der ein Fenster hat, aber ansonsten unsichtbar ist - siehe die Datei coloredlabel.ui.

Ich habe ein "hal_spinbutton"-Widget in Glade definiert und eine Standard-Eigenschaft `value` in der entsprechenden Einstellung festgelegt. Wieso zeigt es Null?

Dies ist auf einen Fehler in der alten Gtk-Version zurückzuführen, die mit Ubuntu 8.04 und 10.04 verteilt wird, und ist wahrscheinlich der Fall für alle Widgets, die Anpassung verwenden. Der Workaround, der zum Beispiel in http://osdir.com/ml/gtk-app-devel-list/2010-04/msg00129.html erwähnt wird, setzt den HAL-Pin-Wert nicht zuverlässig, es ist besser, ihn explizit in einem `on_realize`-Signal-Handler während der Widget-Erstellung zu setzen. Siehe das Beispiel in `configs/apps/gladevcp/by-widget/spinbutton.{ui,py}`.

12.3.10. Fehlersuche

- Stellen Sie sicher, dass Sie die Entwicklungsversion von LinuxCNC installiert haben. Sie brauchen nicht die axisrc Datei nicht mehr, wurde dies in der alten GladeVCP Wiki-Seite erwähnt.
- Starten Sie GladeVCP oder AXIS in einem Terminalfenster. Wenn Sie Python-Fehler erhalten, prüfen Sie, ob neben der neueren `/usr/lib/python2.6/dist-packages/_hal.so` Datei noch eine `/usr/lib/python2.6/dist-packages/_hal.so` Datei herumliegt (den Unterstrich beachten); wenn ja, entfernen Sie die `hal.so` Datei. Sie wurde durch hal.py im selben Verzeichnis ersetzt und verwirrt den Importmechanismus.
- Wenn Sie run-in-place verwenden, führen Sie ein *make clean* aus, um alle versehentlich übrig

gebliebenen hal.so-Dateien zu entfernen, und dann *make*.

- Wenn Sie die Widgets *HAL_table* oder *HAL_HBox* verwenden, beachten Sie bitte, dass sie einen HAL-Pin haben, der standardmäßig ausgeschaltet ist. Dieser Pin steuert, ob die Kinder dieser Container aktiv sind oder nicht.

12.3.11. Implementierungshinweis: Schlüsselbehandlung in AXIS

Wir glauben, dass die Handhabung der Tasten gut funktioniert, aber da es sich um neuen Code handelt, informieren wir Sie darüber, damit Sie auf Probleme achten können; bitte teilen Sie uns Fehler oder seltsames Verhalten mit. Dies ist die Geschichte:

AXIS verwendet den TkInter-Widgetsatz. GladeVCP-Anwendungen verwenden Gtk-Widgets und werden in einem separaten Prozesskontext ausgeführt. Sie werden über das Xembed-Protokoll in AXIS eingebunden. Dies ermöglicht es einer untergeordneten Anwendung wie GladeVCP, sich ordnungsgemäß in ein übergeordnetes Fenster einzufügen und - theoretisch - eine integrierte Ereignisbehandlung zu haben.

Dies setzt jedoch voraus, dass sowohl die übergeordnete als auch die untergeordnete Anwendung das Xembed-Protokoll korrekt unterstützen, was bei Gtk der Fall ist, bei TkInter jedoch nicht. Eine Folge davon ist, dass bestimmte Tasten von einem GladeVCP-Panel nicht unter allen Umständen korrekt an AXIS weitergeleitet werden können. Eine dieser Situationen war der Fall, wenn ein Entry- oder SpinButton-Widget den Fokus hatte: In diesem Fall wurde z.B. eine Escape-Taste nicht an AXIS weitergeleitet und führte zu einem Abbruch, wie es sein sollte, mit möglicherweise katastrophalen Folgen.

Daher werden Tastenereignisse in GladeVCP explizit behandelt und selektiv an AXIS weitergeleitet, um sicherzustellen, dass solche Situationen nicht auftreten können. Für Details siehe die Funktion `keyboard_forward()` in `lib/python/gladevcp/xembed.py`.

12.3.12. Hinzufügen von benutzerdefinierten Widgets

Das LinuxCNC Wiki hat Informationen über das Hinzufügen von benutzerdefinierten Widgets zu GladeVCP. GladeVCP Custom Widgets

12.3.13. GladeVCP-Hilfsanwendungen

Es werden unabhängig installierte GladeVCP-Anwendungen unterstützt, die mit der Platzierung des Systemverzeichnisses übereinstimmen, wie sie von den LINUXCNC_AUX_GLADEVCP- und LINUXCNC_AUX_EXAMPLES-Elementen definiert wird, die vom Skript linuxcnc_var gemeldet werden:

```
$ linuxcnc_var LINUXCNC_AUX_GLADEVCP
/usr/share/linuxcnc/aux_gladevcp
$ linuxcnc_var LINUXCNC_AUX_EXAMPLES
/usr/share/linuxcnc/aux_examples
```

Das durch LINUXCNC_AUX_GLADEVCP definierte Systemverzeichnis (/usr/share/linuxcnc/aux_gladevcp) gibt den Speicherort für eine GladeVCP-kompatible Python-Datei(en) und zugehörige Unterverzeichnisse

an. Die Python-Datei wird beim Start von GladeVCP importiert und für nachfolgende GladeVCP-Anwendungen verfügbar gemacht, einschließlich der eingebetteten Verwendung in unterstützenden GUIs.

Das durch LINUXCNC_AUX_EXAMPLES definierte Systemverzeichnis (/usr/share/linuxcnc/aux_examples) gibt den Speicherort von Beispielkonfigurations-Unterverzeichnissen an, die für Hilfsanwendungen verwendet werden. Siehe den Abschnitt getting-started/running-linuxcnc für *Hinzufügen von Konfigurationsauswahlen*.

Zu Testzwecken kann mit der exportierten Umgebungsvariablen eine Laufzeitspezifikation von Hilfsanwendungen angegeben werden: GLADEVCP_EXTRAS. Diese Variable sollte eine Pfadliste von einem oder mehreren Konfigurationsverzeichnissen sein, die durch ein (:) getrennt sind. Normalerweise wird diese Variable in einer Shell gesetzt, die `linuxcnc` startet, oder im `~/.profile` Startskript eines Benutzers. Beispiel:

```
export GLADEVCP_EXTRAS=~/mygladevcp:/opt/othergladevcp
```

Dateien, die in Verzeichnissen gefunden werden, die mit der Umgebungsvariablen GLADEVCP_EXTRAS angegeben sind, ersetzen gleichnamige Dateien in Unterverzeichnissen des durch LINUXNC_AUX_GLADEVCP angegebenen Systemverzeichnisses (z. B. /usr/share/linuxcnc/aux_gladevcp). Diese Bestimmung ermöglicht es einem Entwickler, eine Anwendung zu testen, indem er GLADEVCP_EXTRAS exportiert, um ein privates Anwendungsverzeichnis anzugeben, ohne ein im System installiertes Anwendungsverzeichnis zu entfernen. Meldungen über abgelehnte Duplikate werden auf stdout ausgegeben.

NOTE Die Unterstützung für GladeVCP-Hilfsanwendungen erfordert ein Python-Modul namens *importlib*. Dieses Modul ist möglicherweise in älteren Installationen wie Ubuntu-Lucid nicht verfügbar.

12.4. GladeVCP-Bibliotheksmodule

Bibliotheken sind vorgefertigte Python-Module, die GladeVCP zusätzliche Funktionen verleihen. Auf diese Weise können Sie auswählen, welche Funktionen Sie wünschen - und müssen die gängigen Funktionen nicht selbst erstellen.

12.4.1. Info

Info ist eine Bibliothek zum Sammeln und Filtern von Daten aus der INI-Datei.

Die verfügbaren Daten und Voreinstellungen:

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
```

```
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")

IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, "share","qtvcp")

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = self.INI.getint("KINS","JOINTS", fallback=0)
AVAILABLE_AXES = ['X','Y','Z']
AVAILABLE_JOINTS = [0,1,2]
GET_NAME_FROM_JOINT = {0:'X',1:'Y',2:'Z'}
GET_JOG_FROM_NAME = {'X':0,'Y':1,'Z':2}
NO_HOME_REQUIRED = False
HOME_ALL_FLAG
JOINT_TYPE = self.INI.getstring(section, "TYPE", fallback="LINEAR")
JOINT_SEQUENCE_LIST
JOINT_SYNC_LIST

JOG_INCREMENTS = None
ANGULAR_INCREMENTS = None
GRID_INCREMENTS

DEFAULT_LINEAR_JOG_VEL = 15 Einheiten pro Minute
MIN_LINEAR_JOG_VEL = 60 Einheiten pro Minute
Länge_LINEAR_JOG_VEL = 300 Einheiten pro Minute

DEFAULT_ANGULAR_JOG_VEL =
MIN_ANGULAR_JOG_VEL =
MAX_ANGULAR_JOG_VEL =

MAX_FEED_OVERRIDE =
MAX_TRAJ_VELOCITY =

AVAILABLE_SPINDLES = self.INI.getint("TRAJ", "SPINDLES", fallback=1)
DEFAULT_SPINDLE_0_SPEED = 200
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_0_OVERRIDE = 100
MIN_SPINDLE_0_OVERRIDE = 50

MAX_FEED_OVERRIDE = 1.5
MAX_TRAJ_VELOCITY

# Benutzer Nachrichten Dialog Info
USRMESS_BOLDTEXT = self.INI.findall("DISPLAY", "MESSAGE_BOLDTEXT")
USRMESS_TEXT = self.INI.findall("DISPLAY", "MESSAGE_TEXT")
USRMESS_TYPE = self.INI.findall("DISPLAY", "MESSAGE_TYPE")
```

```
USRMESS_PINNAME = self.INI.findall("DISPLAY", "MESSAGE_PINNAME")
USRMESS_DETAILS = self.INI.findall("DISPLAY", "MESSAGE_DETAILS")
USRMESS_ICON = self.INI.findall("DISPLAY", "MESSAGE_ICON")
ZIPPED_USRMESS =

self.GLADEVCP = self.INI.find("DISPLAY", "GLADEVCP")

# embedded program info
TAB_NAMES = (self.INI.findall("DISPLAY", "EMBED_TAB_NAME")) or None
TAB_LOCATION = (self.INI.findall("DISPLAY", "EMBED_TAB_LOCATION")) or []
TAB_CMD = (self.INI.findall("DISPLAY", "EMBED_TAB_COMMAND")) or None
ZIPPED_TABS =

MDI_COMMAND_LIST =      (heading: [MDI_COMMAND_LIST], title: MDI_COMMAND")
TOOL_FILE_PATH =        (heading: [EMCIO], title:TOOL_TABLE)
POSTGUI_HALFILE_PATH =  (heading: [HAL], title: POSTGUI_HALFILE)
```

Es gibt einige "Hilfsfunktionen" - hauptsächlich für die Widget-Unterstützung verwendet

```
get_error_safe_setting(self, heading, detail, default=None)
convert_metric_to_machine(data)
convert_imperial_to_machine(data)
convert_9_metric_to_machine(data)
convert_9_imperial_to_machine(data)
convert_units(data)
convert_units_9(data)
get_filter_program(fname)
```

Um diese Module zu importieren, fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from gladevcp.core import Info
```

Um das Modul zu instanziieren, so dass Sie es in einer Handler-Datei verwenden können, fügen Sie diesen Python-Code in Ihren instantiate-Abschnitt ein:

```
###########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
###########################################

INFO = Info()
```

Für den Zugriff auf INFO-Daten verwenden Sie diese allgemeine Syntax:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print('Metric based')
```

12.4.2. Action

Diese Bibliothek wird verwendet, um die Bewegungssteuerung von LinuxCNC zu steuern. Es versucht, zufällige Details zu verbergen und praktische Methoden für Entwickler hinzuzufügen.

Um diese Module zu importieren, fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from gladevcp.core import Action
```

Um das Modul zu instanziieren, damit Sie es verwenden können, fügen Sie den folgenden Python-Code in Ihren instanziierten Abschnitt ein:

```
###########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
###########################################

ACTION = Action()
```

Für den Zugriff auf Aktionsbefehle verwenden Sie eine allgemeine Syntax wie die folgende:

```
ACTION.SET_ESTOP_STATE(state)
ACTION.SET_MACHINE_STATE(state)

ACTION.SET_MACHINE_HOMING(joint)
ACTION.SET_MACHINE_UNHOMED(joint)

ACTION.SET_LIMITS_OVERRIDE()

ACTION.SET_MDI_MODE()
ACTION.SET_MANUAL_MODE()
ACTION.SET_AUTO_MODE()

ACTION.SET_LIMITS_OVERRIDE()

ACTION.CALL_MDI(code)
ACTION.CALL_MDI_WAIT(code)
ACTION.CALL_INI_MDI(number)

ACTION.CALL_OWORD()

ACTION.OPEN_PROGRAM(filename)
ACTION.SAVE_PROGRAM(text_source, fname):

ACTION.SET_AXIS_ORIGIN(axis,value)
ACTION.SET_TOOL_OFFSET(axis,value,fixture = False)

ACTION.RUN()
ACTION.ABORT()
ACTION.PAUSE()
```

```
ACTION.SET_MAX_VELOCITY_RATE(rate)
ACTION.SET_RAPID_RATE(rate)
ACTION.SET_FEED_RATE(rate)
ACTION.SET_SPINDLE_RATE(rate)

ACTION.SET_JOG_RATE(rate)
ACTION.SET_JOG_INCR(incr)
ACTION.SET_JOG_RATE_ANGULAR(rate)
ACTION.SET_JOG_INCR_ANGULAR(incr, text)

ACTION.SET_SPINDLE_ROTATION(direction = 1, rpm = 100, number = 0)
ACTION.SET_SPINDLE_FASTER(number = 0)
ACTION.SET_SPINDLE_SLOWER(number = 0)
ACTION.SET_SPINDLE_STOP(number = 0)

ACTION.SET_USER_SYSTEM(system)

ACTION.ZERO_G92_OFFSET()
ACTION.ZERO_ROTATIONAL_OFFSET()
ACTION.ZERO_G5X_OFFSET(num)

ACTION.RECORD_CURRENT_MODE()
ACTION.RESTORE_RECORDED_MODE()

ACTION.SET_SELECTED_AXIS(jointnum)

ACTION.DO_JOG(jointnum, direction)
ACTION.JOG(jointnum, direction, rate, distance=0)

ACTION.TOGGLE_FLOOD()
ACTION.SET_FLOOD_ON()
ACTION.SET_FLOOD_OFF()

ACTION.TOGGLE_MIST()
ACTION.SET_MIST_ON()
ACTION.SET_MIST_OFF()

ACTION.RELOAD_TOOLTABLE()
ACTION.UPDATE_VAR_FILE()

ACTION.TOGGLE_OPTIONAL_STOP()
ACTION.SET_OPTIONAL_STOP_ON()
ACTION.SET_OPTIONAL_STOP_OFF()

ACTION.TOGGLE_BLOCK_DELETE()
ACTION.SET_BLOCK_DELETE_ON()
ACTION.SET_BLOCK_DELETE_OFF()

ACTION.RELOAD_DISPLAY()
ACTION.SET_GRAPHICS_VIEW(view)

ACTION.UPDATE_MACHINE_LOG(text, option=None):

ACTION.SET_DISPLAY_MESSAGE(string)
ACTION.SET_ERROR_MESSAGE(string)
```

Es gibt einige *Hilfsfunktionen*, die hauptsächlich für die Unterstützung dieser Bibliothek verwendet werden

```
get_jog_info (num)
jnum_check(num)
ensure_mode(modes)
open_filter_program(filename, filter)
```

12.5. GladeVCP mitgelieferte Panels

GladeVCP kann zur **Erstellung von Bedienfeldern** verwendet werden, die mit *HAL* und/oder dem motion controller verbunden sind.

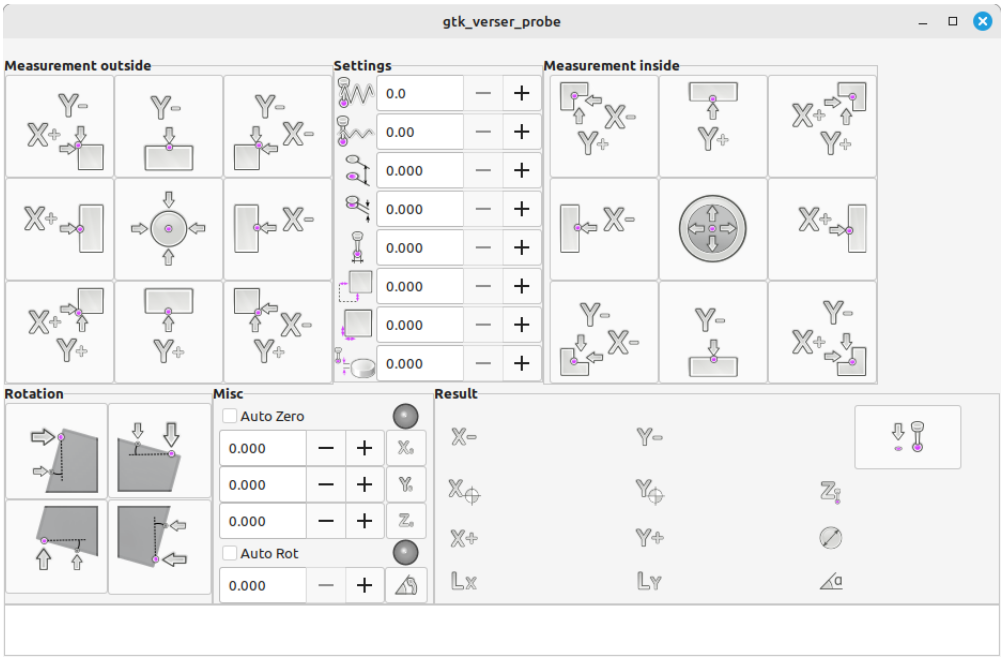
12.5.1. Eingebaute virtuelle Bedienpulte (engl. virtual control panels)

Es sind mehrere **integrierte Panels** verfügbar.

Geben Sie in einem Terminal `gladevcp` ein, um eine Liste zu sehen.

GTK Verser Taster (engl. probe)

A GTK based version of the third party Verser probe.



This is a version from 2015 by Serguei Glavatski which has **less functionality** than the current, but it **takes up less space on the screen** (no DRO e.g.). This can be useful on smaller displays.

For Reference - The Current Version:

The documentation and source code of the current version can be found here:

- https://vers.ge/en/blog/useful-articles/probe-screen-v28

- https://github.com/verser-git/probe_screen_v2.9

Modification of the INI file to use as embedded panel in Gmoccapy:

```
[DISPLAY]
DISPLAY = gmoccapy
EMBED_TAB_NAME = Probe
EMBED_TAB_LOCATION = ntb_user_tabs
EMBED_TAB_COMMAND = gladevcp -x {XID} gtk_verser_probe

[TOOLSENSOR]
RAPID_SPEED = 600

[RS274NGC]
# for package install:
SUBROUTINE_PATH = ./macros:/usr/share/linuxcnc/nc_files/gtk_probe/

# For RIP installation, use the path according to your directory:
# SUBROUTINE_PATH = ./macros:~/linuxcnc/nc_files/probe/gtk_probe/
```

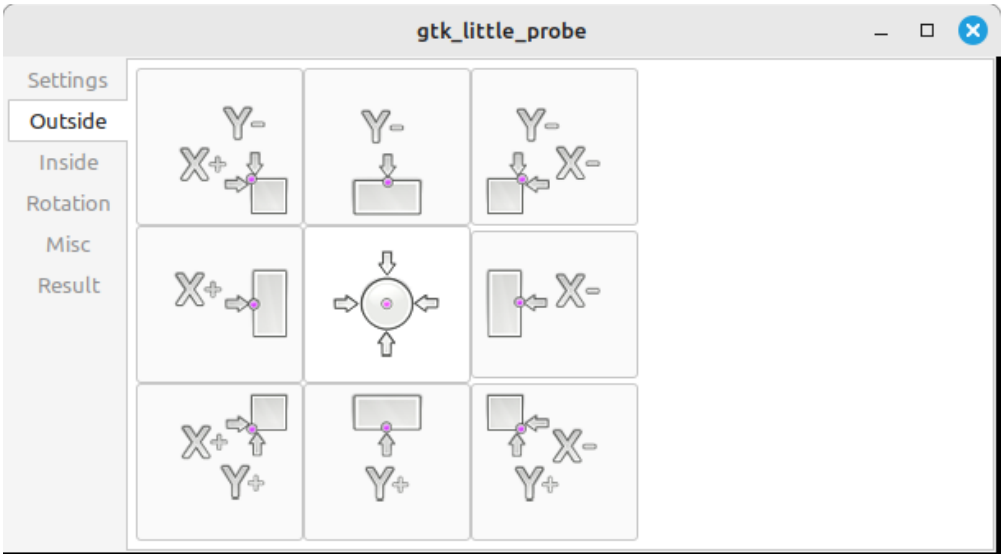
Example using dbounce with a Mesa card (HAL file):

```
# ---probe signal---
loadrt dbounce names=dbounce.probe
addf dbounce.probe   servo-thread

setp dbounce.probe.delay 5
net probe-db  dbounce.probe.in    <=  hm2_7i96.0.gpio.000.in
net probe-in  motion.probe-input  <=  dbounce.probe.out
```

GTK Little Probe

A GTK based version of the third party Verser probe, modified for a tabbed layout for smaller displays.



This is a modification of the 2015 version of GTK Verser Probe. It moves the elements from one window to containers with switchable vertical tabs. The goal was to reduce the window size.

Modification of the INI file to use as embedded panel in Gmoccapy:

```
[DISPLAY]
DISPLAY = gmoccapy
EMBED_TAB_NAME = Probe
EMBED_TAB_LOCATION = ntb_preview
EMBED_TAB_COMMAND = gladevcp -x {XID} gtk_little_probe

[TOOLSENSOR]
RAPID_SPEED = 600

[RS274NGC]
# for package install:
SUBROUTINE_PATH = ./macros:/usr/share/linuxcnc/nc_files/probe/gtk_probe/

# For RIP installation, use the path according to your directory:
# SUBROUTINE_PATH = ./macros:~/linuxcnc/nc_files/probe/gtk_probe/
```

Use only one such Probe Screen in one LCNC configuration.

Example using dbounce with a Mesa card (HAL file):

```
# ---probe signal---
loadrt dbounce names=dbounce.probe
addf dbounce.probe   servo-thread

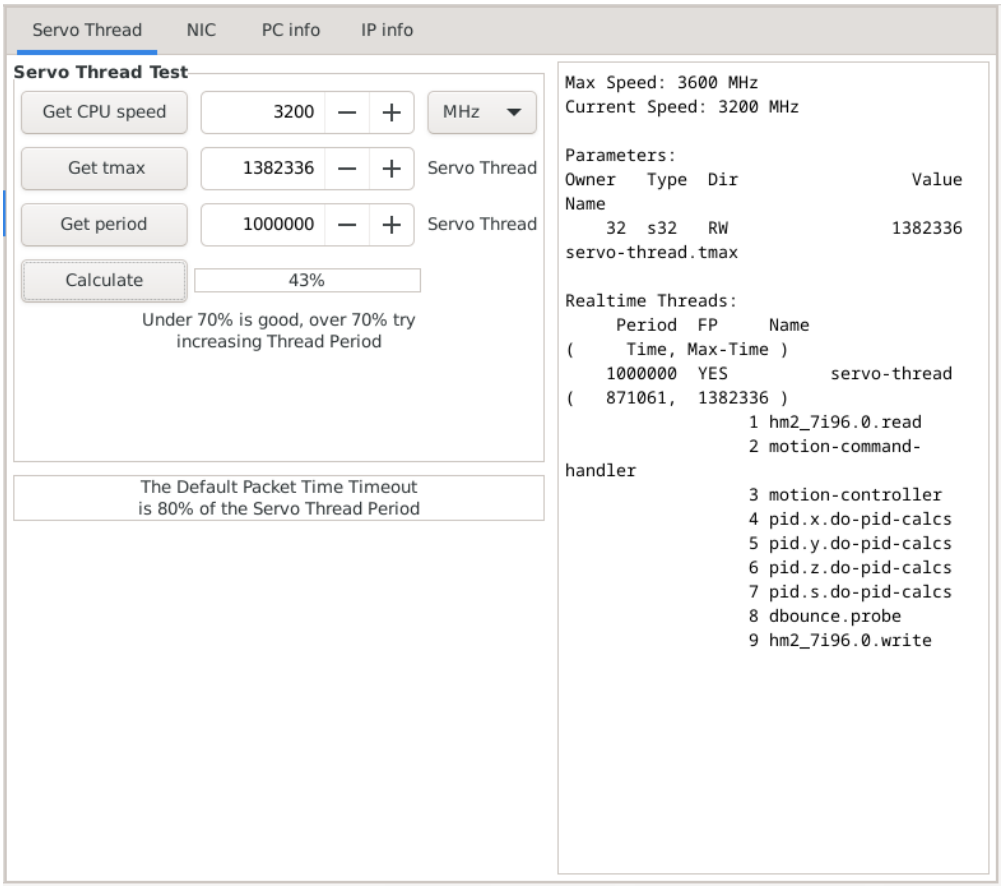
setp dbounce.probe.delay 5
net probe-db  dbounce.probe.in    <=  hm2_7i96.0.gpio.000.in
net probe-in  motion.probe-input  <=  dbounce.probe.out
```

GTK Mesa Tests

This screen allows the user to verify whether they have a suitable and well-tuned PC for their Mesa card.

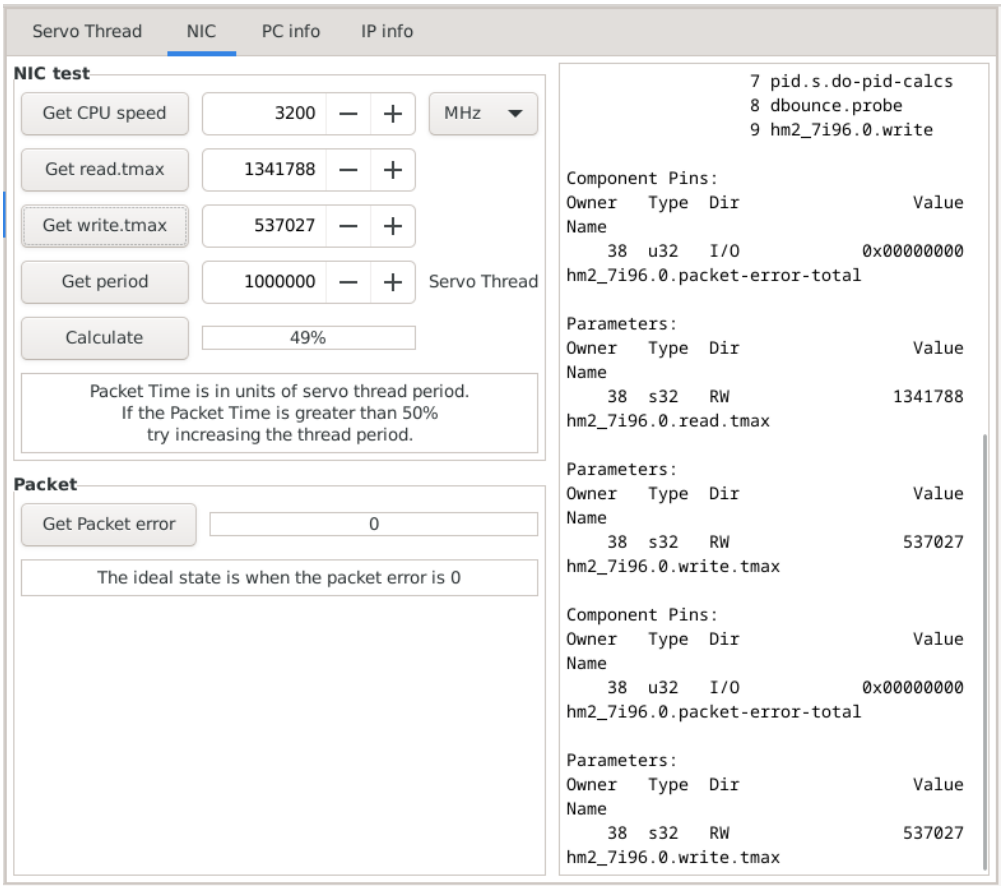
Based on the design of Mesa Configuration Tool II https://github.com/jethornton/mesact Copyright (c) 2022 jethornton

Although this screen was originally part of the configurator, it is not the configurator. It is a control system. Mesa tests uses data from a running LinuxCNC, but does not send any data to LinuxCNC, nor does it set anything. Parameter adjustment is only allowed to simulate the desired state.



Servo Thread-Test:

- Drücken Sie den "Get CPU speed" Button.
- Drücken Sie den Button "Get tmax"
- Drücken Sie den "Get period" Button.
- Drücken Sie den "Calculate" Button.
- überprüfen des Ergebnisses [%]



NIC-Test:

- Drücken Sie den "Get CPU speed" Button.
- Drücken Sie den "Get read.tmax" Button.
- Drücken Sie den "Get write.tmax" Button.
- Drücken Sie den "Get period" Button.
- Drücken Sie den "Calculate" Button.
- überprüfen des Ergebnisses [%]

Mesa Tests is a tool designed for beginners to sleep better or to know if they need to buy better hardware or tune their hardware better. It is better to use tools like Halshow or Halscope to monitor the parameters (pins,thread, ...) used. The list of parameters is displayed on the right side of the screen.

12.6. QtVCP

QtVCP ist eine **Infrastruktur zum Erstellen von benutzerdefinierten CNC-Bildschirmen oder Bedienfeldern für LinuxCNC**.

Es zeigt eine *`.ui`-Datei an, die mit dem Qt Designer*-Bildschirmeditor erstellt wurde, und kombiniert diese mit *Python-Programmierung*, um einen GUI-Bildschirm für den Betrieb einer CNC-Maschine zu erstellen.

QtVCP ist vollständig *anpassbar*: Sie können verschiedene Schaltflächen und Status-LEDs usw. hinzufügen oder Python-Code für eine noch feinere Anpassung einfügen.

12.6.1. Schaukasten

Einige Beispiele für mit QtVCP erstellte Bildschirme und virtuelle Bedienfelder:

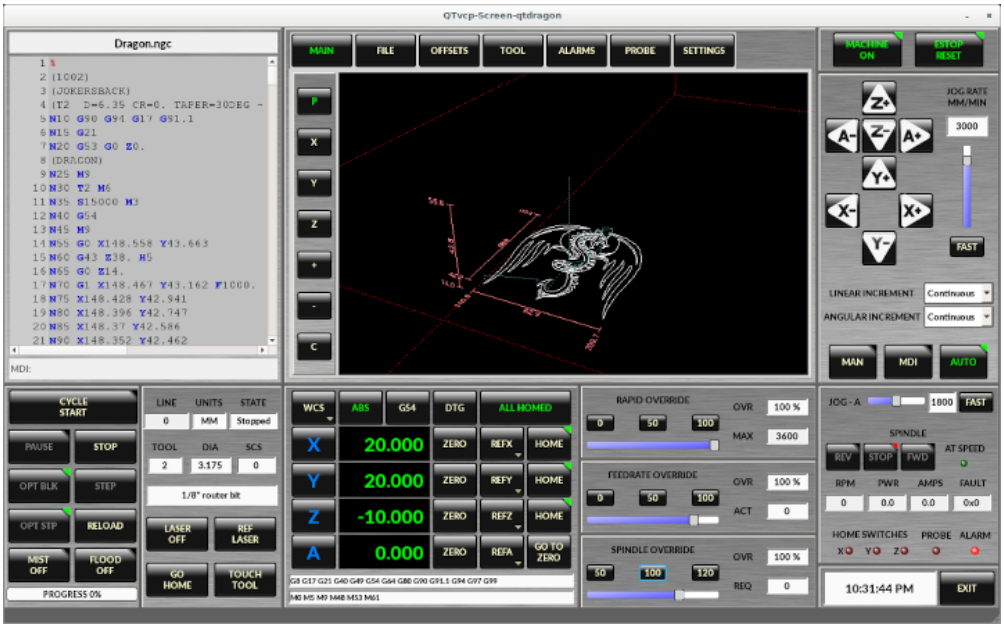


Figure 275. QtDragon - 3/4-Achsen-Beispiel

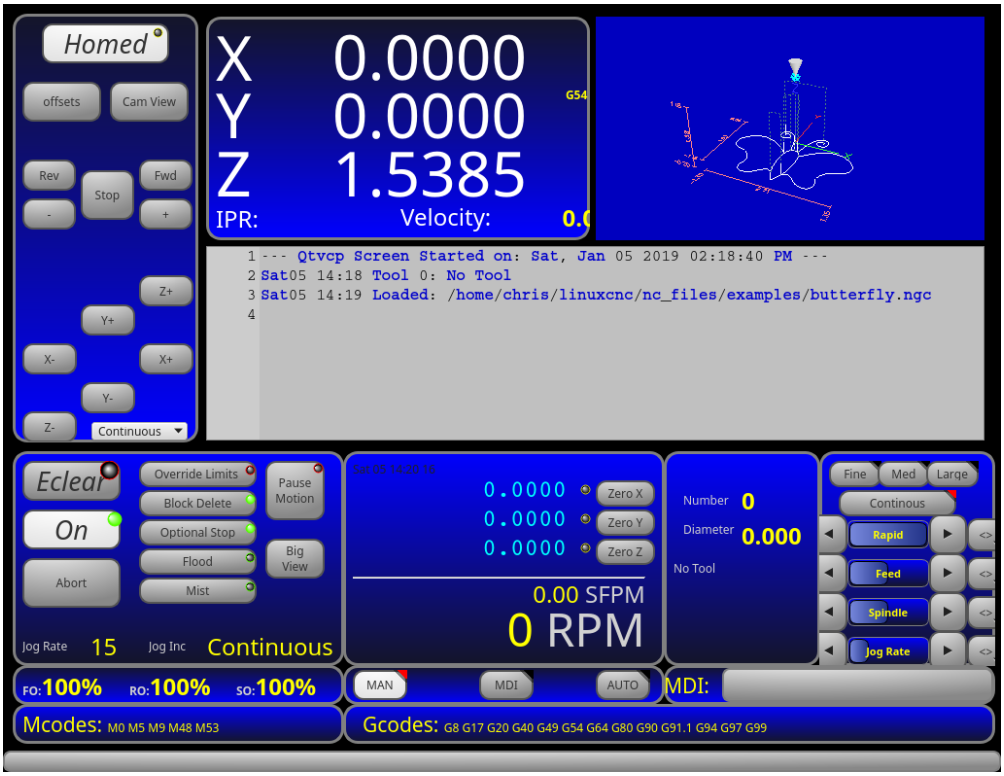


Figure 276. QtDefault - 3-Achsen-Beispiel

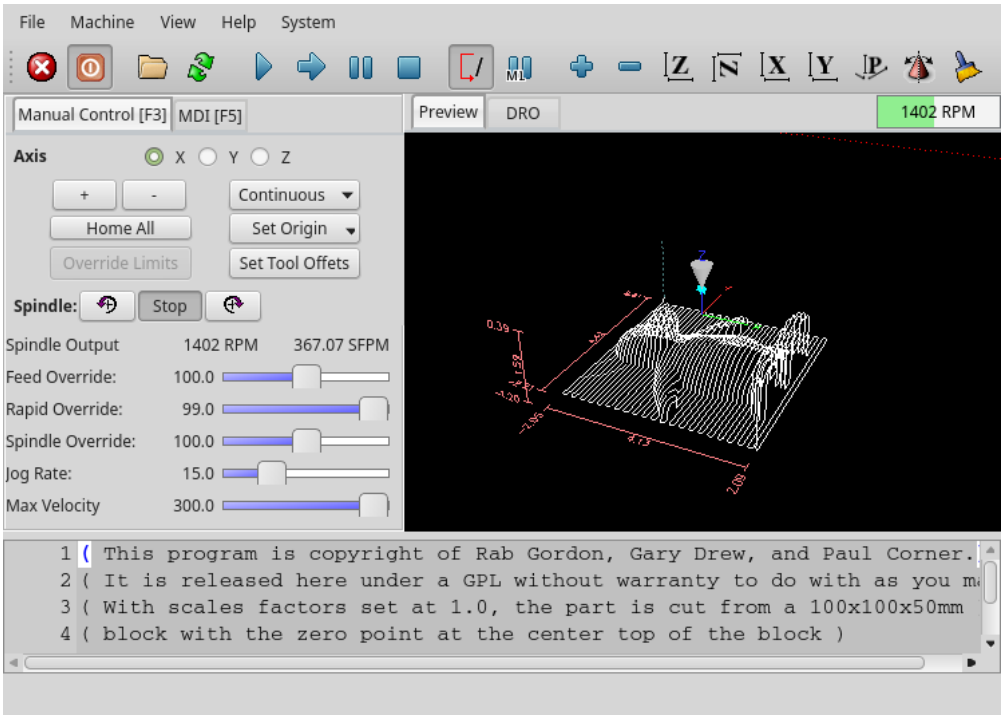


Figure 277. QtAxis - Beispiel für selbsteinstellende Achsen

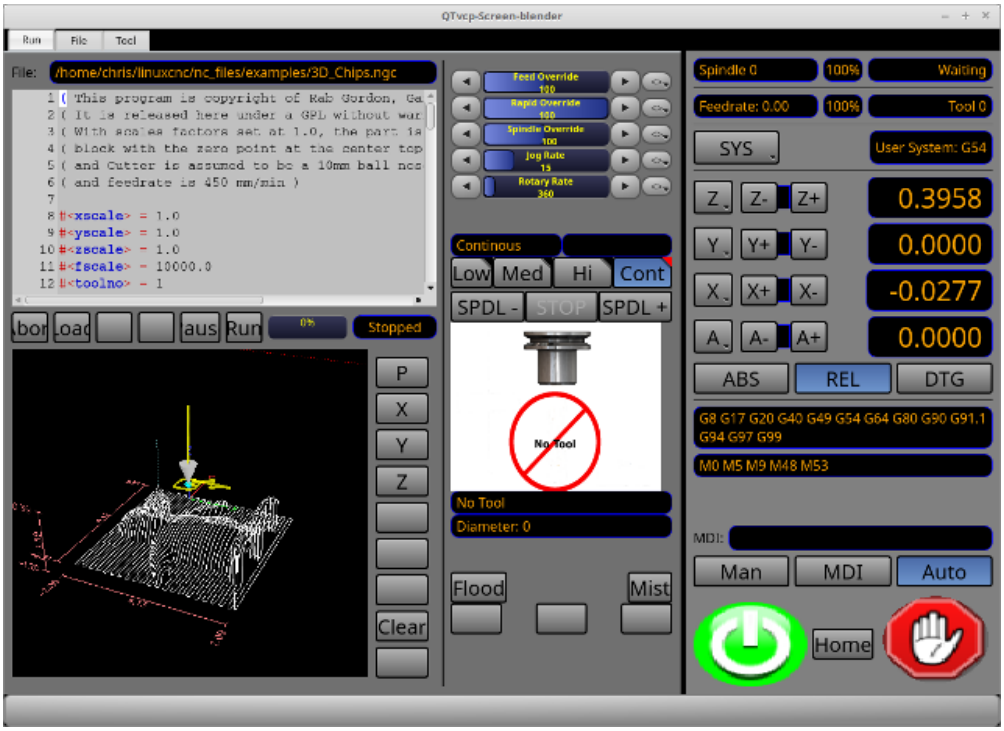


Figure 278. Blender - 4-Achsen-Beispiel

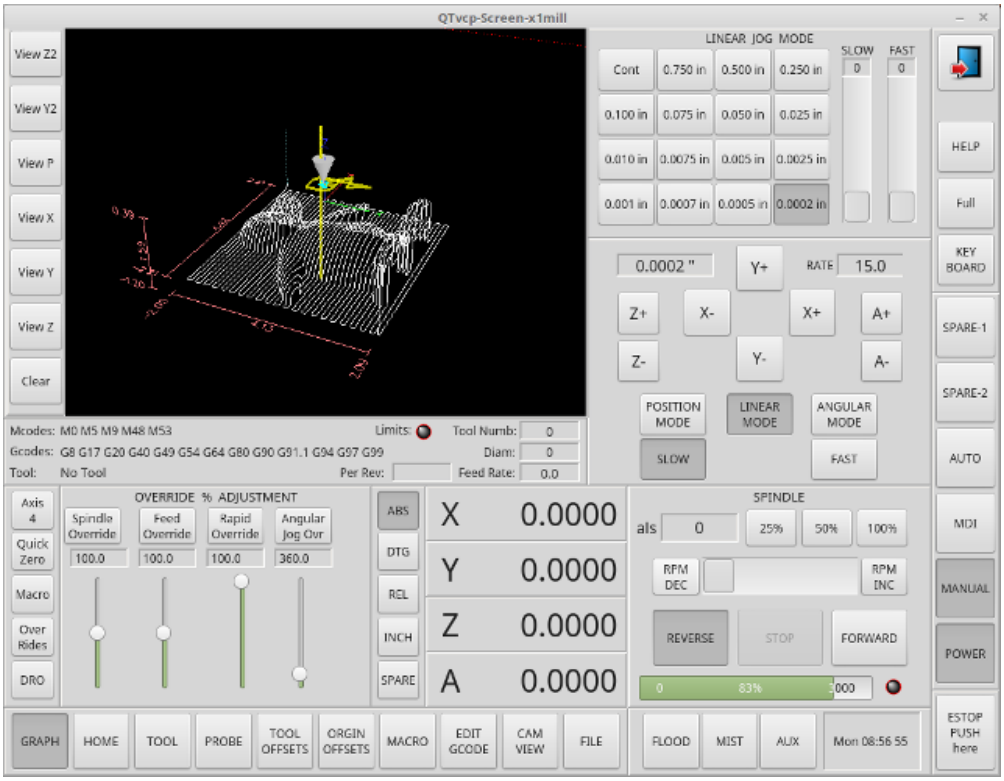


Figure 279. X1mill - 4-Achsen-Beispiel

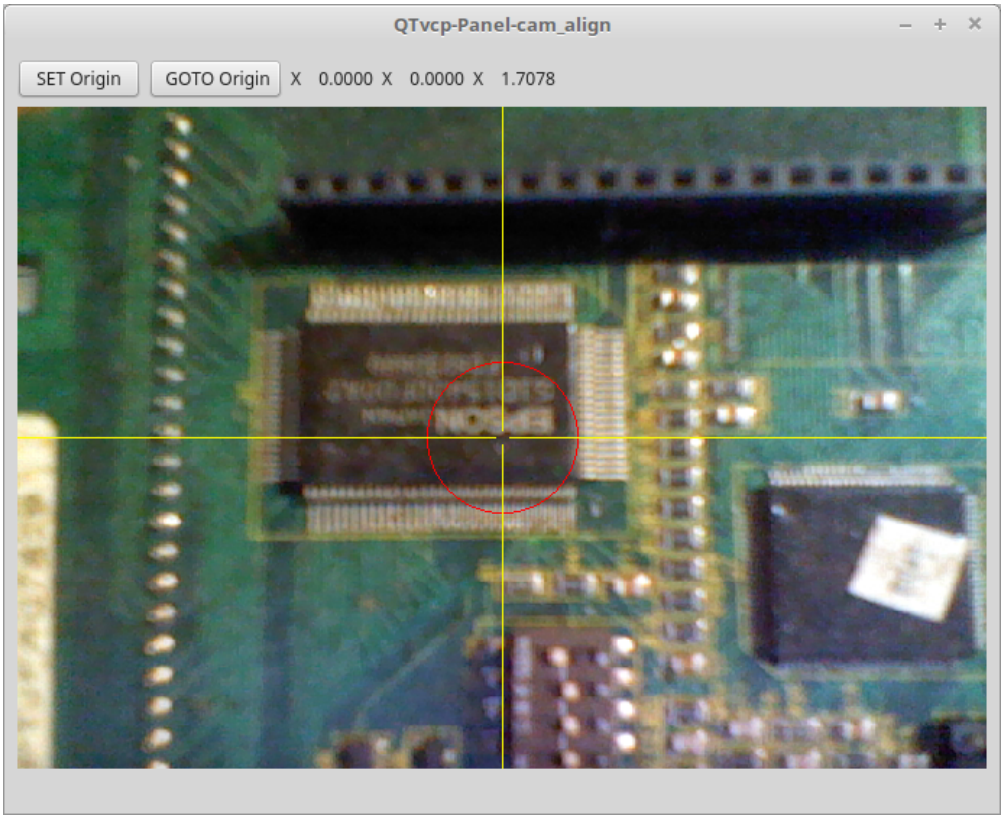


Figure 280. cam_align – Kameraausrichtung VCP

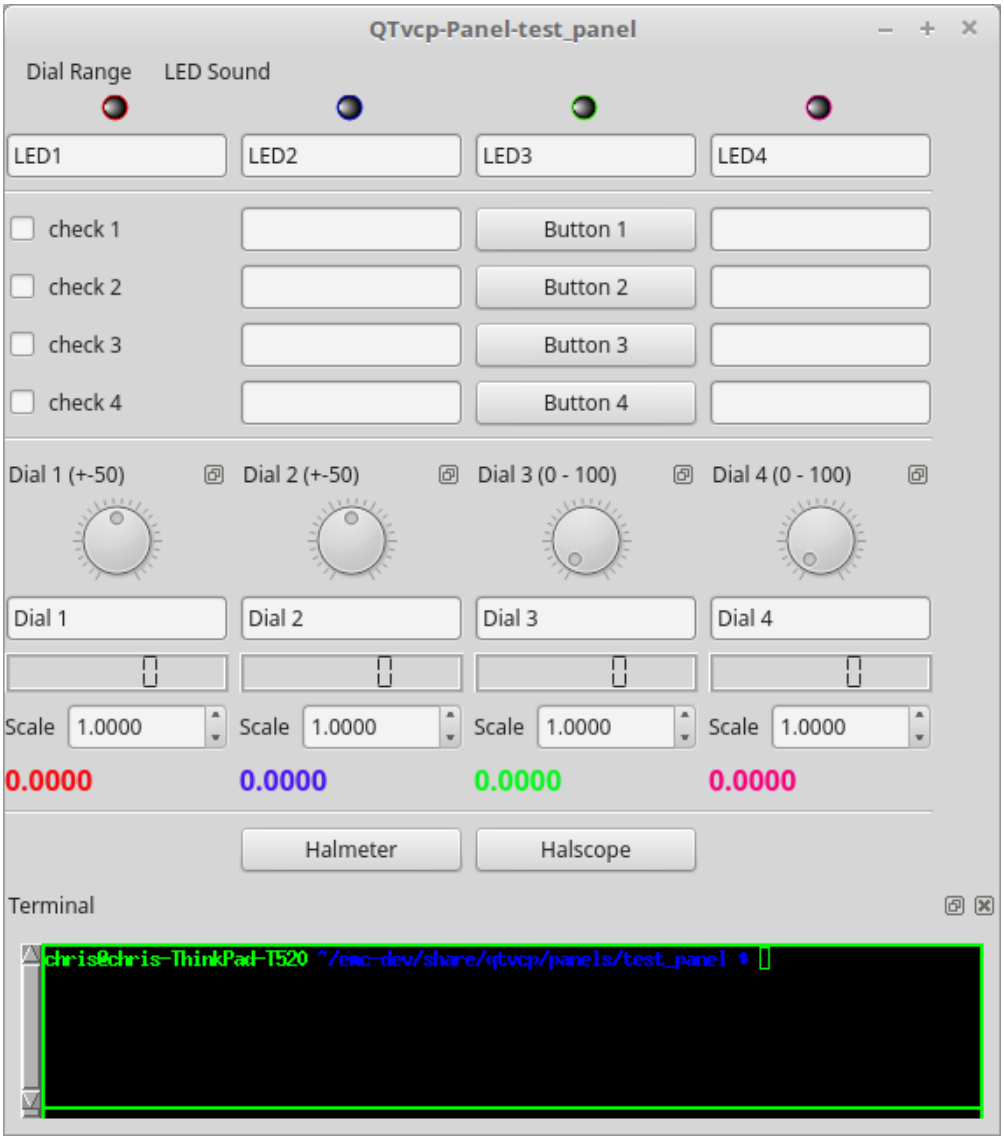


Figure 281. test_panel - Test Panel VCP

12.6.2. Übersicht

Zwei Dateien werden, einzeln oder in Kombination, verwendet, um Anpassungen vorzunehmen:

- Eine **UI-Datei**, bei der es sich um eine *XML*-Datei handelt, die mit dem grafischen Editor *Qt Designer* erstellt wurde.
- Eine **Handler-Datei**, die eine Textdatei mit *Python*-Code ist.

Normalerweise verwendet QtVCP die standardmäßige UI- und Handler-Datei, aber Sie können QtVCP so einstellen, dass es "lokale" UI- und Handler-Dateien verwendet.
Eine ***lokale* Datei** ist eine Datei, die sich im *Konfigurationsordner* befindet, der den Rest der Anforderungen des Rechners definiert.

Man ist nicht darauf beschränkt, ein benutzerdefiniertes Panel auf der rechten Seite oder eine benutzerdefinierte Registerkarte hinzuzufügen, da QtVCP den *Qt Designer* (den Editor) und *PyQt5* (das Widget-Toolkit) nutzt.

QtVCP hat einige **spezielle LinuxCNC Widgets und Aktionen** hinzugefügt.

Es gibt spezielle Widgets, um Widgets von Drittanbietern mit HAL-Pins zu verbinden.
Es ist möglich, Widget-Antworten zu erstellen, indem man Signale mit Python-Code in der Handler-Datei verbindet.

QtVCP Widgets

QtVCP nutzt die **PyQt5-Toolkits** für die Einbeziehung von LinuxCNC.

Widget ist der *allgemeine Name für die Benutzerschnittstellen-Objekte* wie Buttons und Beschriftungen in PyQt5.

Es steht Ihnen frei, alle verfügbaren **Standard-Widgets** im Qt Designer-Editor zu verwenden.

Es gibt auch **spezielle Widgets** für LinuxCNC gemacht, dass die Integration einfacher zu machen.
Diese sind in drei Überschriften auf der linken Seite des Editors aufgeteilt:

- Eine ist *nur für HAL Widgets*;
- Eine ist für *CNC-Steuerungs-Widgets*;
- Eine ist für *Dialog-Widgets*.

Es steht Ihnen frei, sie auf Ihrer Tafel beliebig zu mischen.

Ein sehr wichtiges Widget für die CNC-Steuerung ist das **`ScreenOptions`-Widget**: Es fügt dem Bildschirm nichts Visuelles hinzu, sondern ermöglicht die Auswahl wichtiger Details, die dann in der Handler-Datei kodiert werden müssen.

INI-Einstellungen

Wenn Sie QtVCP zur Erstellung eines CNC-Bewegungssteuerungsbildschirms (und nicht eines HAL-basierten Panels) verwenden, fügen Sie in der INI-Datei im Abschnitt `[DISPLAY]` eine Zeile mit folgendem Muster ein:

```
DISPLAY = qtvcp <Optionen> <Bildschirmname>
```

NOTE Alle "<Optionen>" müssen vor "<Bildschirmname>" stehen.

Optionen

- `-d` Debugging an.
- `-i` Infoausgabe aktivieren.
- `-v` Aktiviert die ausführliche Debug-Ausgabe.
- `-q` Aktiviert nur die Fehler-Debug-Ausgabe.
- `-a` Fenster immer in Vordergrund (engl. top) setzen.
- `-c NAME` Name der HAL-Komponente. Standardmäßig wird der UI-Dateiname verwendet.
- `-g GEOMETRIE`Legt die Geometrie WIDTHxHEIGHT+XOFFSET+YOFFSET fest. Die Werte sind in Pixel-Einheiten, XOFFSET/YOFFSET wird vom linken oberen Bildschirmrand

aus referenziert. Verwenden Sie -g WIDTHxHEIGHT, um nur die Größe zu bestimmen, oder -g +XOFFSET+YOFFSET, um nur die Position zu bestimmen. Beispiel: `-g 200x400+0+100

- `-H DATEI`Führt HAL-Anweisungen aus DATEI mit halcmd aus, nachdem die Komponente eingerichtet und bereit ist.
- `-m` Fenster maximieren.
- `-f` Vollbild des Fensters.
- `-t THEME` Standard ist das Systemdesign
- `-x XID` Einbindung in ein X11-Fenster, das nicht die Integration unterstützt.
- `--push_xid` Sendet die X11-Fenster-Identifikationsnummer von QtVCP an die Standardausgabe; zum Einbetten.
- `-u USERMOD` Dateipfad einer Ersatz-Handler-Datei.
- `-o USEROPTS` Übergibt einen String an die Handler-Datei von QtVCP unter der Listenvariablen `self.w.USEROPTIONS_`. Können mehrere -o sein.
- `-force_pyqt=6` Forces QtVCP to use PyQt6, if it's availble, otherwise uses PyQt5

<Bildschirm_name>

`<Bildschirmname>` ist der *Basisname der .ui und _handler.py Dateien*. Wenn `<Bildschirmname>` fehlt, wird der Standardbildschirm geladen.

QtVCP nimmt an, dass die UI-Datei und die Handler-Datei den **gleichen Basisnamen** verwenden. QtVCP sucht zunächst im LinuxCNC-Konfigurationsverzeichnis, das gestartet wurde, nach den Dateien, dann im System-Skin-Ordner mit den Standardbildschirmen.

Zykluszeiten

```
[DISPLAY]
CYCLE_TIME = 100
GRAPHICS_CYCLE_TIME = 100
HALPIN_CYCLE = 100
```

Stellt die Reaktionsgeschwindigkeit der GUI-Aktualisierungen in Millisekunden ein. Standardwert ist 100, nutzbarer Bereich 50 - 200.

Die Widgets, Grafiken und die HAL-Pin-Aktualisierung können separat eingestellt werden.

Wenn die Aktualisierungszeit nicht richtig eingestellt ist, kann der Bildschirm nicht mehr reagieren oder stark ruckeln.

Qt Designer UI Datei

Eine Qt Designer-Datei ist eine Textdatei, die im *XML*-Standard organisiert ist und das **Layout und die Widgets** des Bildschirms beschreibt.

PyQt5 verwendet diese Datei, um die Anzeige zu erstellen und auf diese Widgets zu reagieren.

Der Qt Designer Editor macht es relativ einfach, diese Datei zu erstellen und zu bearbeiten.

Handler-Dateien

Eine Handler-Datei ist eine Datei, die *Python*-Code enthält, der **zu den QtVCP-Standardroutinen hinzugefügt wird**.

Eine Handler-Datei erlaubt es, Voreinstellungen zu *ändern* oder einem QtVCP-Bildschirm *Logik hinzuzufügen*, ohne den Kerncode von QtVCP zu verändern. Auf diese Weise können Sie **eigene Verhaltensweisen** implementieren.

Falls vorhanden, wird eine Handler-Datei geladen. **Es ist nur eine Datei** erlaubt.

Bibliotheken Module

QtVCP, so wie es gebaut ist, tut wenig mehr als den Bildschirm anzuzeigen und auf Widgets zu reagieren. Für weitere **vorgefertigte Verhaltensweisen** gibt es verfügbare Bibliotheken (zu finden in `lib/python/qtvcp/lib` in RIP LinuxCNC install).

Bibliotheken (engl. und neudeutsch **libraries**) sind vorgefertigte Python-Module, die QtVCP **zusätzliche Funktionen** verleihen. Auf diese Weise können Sie auswählen, welche Funktionen Sie wünschen - und müssen die üblichen nicht selbst erstellen.

Zu diesen Bibliotheken gehören:

- `audio_player`
- `aux_program_loader`
- `keybindings`
- `message`
- `preferences`
- `notify`
- `virtual_keyboard`
- `machine_log`

Themen

Designs sind eine Möglichkeit, das ***look and feel*** der Widgets auf dem Bildschirm zu ändern.

Zum Beispiel kann die *Farbe* oder *Größe* von Schaltflächen und Schiebereglern mit Hilfe von Themen geändert werden.

Das *Windows-Thema* ist der Standard für Bildschirme.
Das *Systemthema* ist der Standard für Bedienfelder.

Um die verfügbaren Themen zu sehen, können Sie sie mit dem folgenden Befehl in einem Terminal laden:

```
qtvcp -d -t <theme_name>
```

QtVCP kann auch mit *Qt-Stylesheets (QSS)* unter Verwendung von CSS angepasst werden.

Lokale Dateien

Falls vorhanden, werden die lokalen UI/QSS/Python-Dateien im Konfigurationsordner anstelle der Standard-UI-Dateien geladen.

Lokale UI/QSS/Python-Dateien ermöglichen es Ihnen, Ihre eigenen Designs anstelle der Standardbildschirme zu verwenden.

QtVCP sucht nach einem Ordner mit dem Namen <*screen_name*> (im Ordner für die Startkonfiguration, der die INI-Datei enthält).

In diesem Ordner lädt QtVCP jede der folgenden Dateien:

- `_<screen_name>_.ui`,
- `<screen_name>_handler.py`, und
- `_<screen_name>_.qss`.

Veränderung mitglieferter Bildschirmmasken

Es gibt *drei Möglichkeiten*, einen Bildschirm/Panel anzupassen.

Kleinere Stylesheet-Änderungen

Stylesheets können zum **Setzen von Qt-Eigenschaften** verwendet werden. Wenn ein Widget Eigenschaften verwendet, können diese normalerweise durch Stylesheets verändert werden.

Beispiel für ein Widget mit den zugehörigen Style-Sheet-Einstellungen.

```
State_LED #name_of_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
  }
```

Handler Patching – Unterklassenbildung eingebauter Bildschirme

QtVCP kann eine untergeordnete Version der Standard-Handler-Datei laden. In dieser Datei können die ursprünglichen Funktionen verändert oder neue hinzugefügt werden.
Unterklassifizierung bedeutet lediglich, dass unsere Handler-Datei zuerst die originale Handler-Datei lädt und dann unseren neuen Code darüberlegt - im Wesentlichen eine Art Patch.
Dies ist nützlich, um Verhalten zu ändern oder zu erweitern, während weiterhin Updates der Standard-Handler aus den LinuxCNC-Repositories übernommen werden.

Möglicherweise muss dennoch der **Handler Copy Dialog** verwendet werden, um die original Handler-

Datei zu kopieren und zu entscheiden, wie sie gepatcht wird. Siehe **Custom Handler File**.

Es sollte einen Ordner im Konfigurationsordner geben; für Bildschirme: benannt als *<KONFIGURATIONSORDNER>/qtvcp/screens/<BILDSCHIRMNAME>/*
Fügen Sie dort die Handler-Patch-Datei hinzu, benannt wie folgt <ORIGINAL BILDSCHIRMNAME>_handler.py,
d.h. für QtDragon würde die Datei *qtdragon_handler.py* genannt werden.

Hier ist ein Beispiel, um X-Achse Jog-Pins zu einem Bildschirm wie QtDragon hinzuzufügen:

```
import sys
import importlib
from qtvcp.core import Path, Qhal, Action
PATH = Path()
QHAL = Qhal()
ACTION = Action()

# besorge Referenz zu Original Handler Datei, um sie zu spezialisieren (engl. subclass)
sys.path.insert(0, PATH.SCREENDIR)
module = "{}.{}_handler".format(PATH.BASEPATH,PATH.BASEPATH)
mod = importlib.import_module(module, PATH.SCREENDIR)
sys.path.remove(PATH.SCREENDIR)
HandlerClass = mod.HandlerClass

# return our subclassed handler object to QtVCP
def get_handlers(halcomp, widgets, paths):
    return [UserHandlerClass(halcomp, widgets, paths)]

# Unterklasse HandlerClass die oben import wurde
class UserHandlerClass(HandlerClass):
    # Ausgabe einer Nachricht auf dem Terminal, damit wir wissen, dass sie geladen wurde
    print('\nCustom subclassed handler patch loaded.\n')

    def init_pins(self):
        # Aufruf der original Handler init_pins Funktionen
        super().init_pins()

        # add jog pins X axis
        pin = QHAL.newpin("jog.axis.jog-x-plus", QHAL.HAL_BOOL, QHAL.HAL_IN)
        pin.value_changed.connect(lambda s: self.kb_jog(s, 0, 1, fast = False, linear =
True))

        pin = QHAL.newpin("jog.axis.jog-x-minus", QHAL.HAL_BOOL, QHAL.HAL_IN)
        pin.value_changed.connect(lambda s: self.kb_jog(s, 0, -1, fast = False, linear =
True))
```

Kleinere Änderungen am Python-Code

Eine weitere Python-Datei kann verwendet werden, um Befehle auf dem Bildschirm **hinzuzufügen**, nachdem die Handler-Datei geparst wurde. Dies kann für kleinere Änderungen nützlich sein, während die Standard-Handler-Updates aus den LinuxCNC-Repositorien weiterhin berücksichtigt werden.

NOTE	Handler Patching ist ein besserer Weg, um Änderungen hinzuzufügen - Instanz-Patching

> ist schwarzer magischer voodoo - dies ist hier nur der Nostalgie halber dokumentiert.

In der *INI Datei* unter der `[DISPLAY] ` Überschrift fügen Sie hinzu *`USER_COMMAND_FILE = _PATH_*
PATH kann jeder gültige Pfad sein. Er kann `~` für das Heimatverzeichnis oder `WORKINGDIRECTORY` oder `CONFIGDIRECTORY` verwenden, um QtVCPs Vorstellung von diesen Verzeichnissen zu repräsentieren, z. B.:

```
[DISPLAY]
USER_COMMAND_FILE = CONFIGFOLDER/<Bildschirm_name_hinzugefuegte_Befehle>
```

Wenn kein Eintrag in der *INI* gefunden wird, sucht QtVCP im **Standardpfad**. Der Standardpfad befindet sich im Konfigurationsverzeichnis als versteckte Datei mit dem Basisnamen des Bildschirms und rc, d.h.: **`CONFIGURATION DIRECTORY/.<Bildschirmname>rc`**.

Diese Datei wird als Python-Code im **handler-Dateikontext** gelesen und ausgeführt.

Nur lokale Funktionen und lokale Attribute können referenziert werden. +
Globale Bibliotheken, die in der Handler-Datei des Bildschirms definiert sind, können referenziert werden durch Importieren der handler-Datei.
Diese werden normalerweise vollständig mit Großbuchstaben ohne vorangestelltes *self* dargestellt. +
self verweist auf Funktionen der window (engl. für Fenster) Klasse (engl. class). +
self.w verweist typischerweise auf die Widgets (engl. Kunstwort für Fenster-Elemente).

Was verwendet werden kann, mag je nach Bildschirm und Entwicklungszyklus variieren.

Ein einfaches Beispiel

Verweis auf das Hauptfenster, um den Titel zu ändern (wird nicht angezeigt, wenn INI-Einträge für die Titeländerung verwendet werden).

```
self.w.setWindowTitle('Mein Titel-Test')
```

Ein fortgeschrittenes Beispiel für das Patchen von Instanzen

Dies könnte mit der Handler-Datei des QtDragon-Bildschirms funktionieren.
Hier zeigen wir, wie man neue Funktionen hinzufügt und bestehende überschreibt.

```
# Benötigt für Instanz-Patch
# Referenz: https://ruivieira.dev/python-monkey-patching-for-readability.html
import types

# importiere das Handlerfile, um einen Verweis auf dessen Bibliotheken zu erhalten.
# benutze <Bildschirmname>_handler
import qtdragon_handler as hdlr

# Dies ist eigentlich eine unbeschränkte Funktion mit 'obj' als Parameter.
# Sie rufen diese Funktion ohne das übliche vorangestellte 'self' auf.
# Das liegt daran, dass sie nicht in die ursprüngliche Instanz der Handler-Klasse
eingefügt wird.
# Sie wird nur von Code in dieser Datei aufgerufen.
```

```
def test_function(obj):
    print(dir(obj))

# Dies ist eine neue Funktion, die wir der bestehenden Handler-Klasseninstanz hinzufügen
werden.
# Beachten Sie, dass sie die unbeschränkte Funktion mit 'self' als Parameter aufruft,
'self' ist der einzige verfügbare globale Referenz.
# Sie verweist auf die Fensterinstanz
def on_keycall_F10(self,event,state,shift,cntrl):
    if state:
        print ('F10')
        test_function(self)

# Dies wird verwendet, um eine bestehende Funktion in der bestehenden Handler-
Klasseninstanz außer Kraft zu setzen.
# Beachten Sie, dass wir auch eine Kopie der ursprünglichen Funktion aufrufen.
# Dies zeigt, wie man eine bestehende Funktion um zusätzliche Funktionen erweitert.
def on_keycall_F11(self,event,state,shift,cntrl):
    if state:
        self.on_keycall_F11_super(event,state,shift,cntrl)
        print ('Hallo')

# Wir verweisen auf die KEYBIND-Bibliothek, die in der ursprünglichen Handler-
Klasseninstanz instanziiert wurde
# durch Hinzufügen von 'hdlr.' (vom imp).
# Diese Funktion weist KEYBIND an, 'on_keycall_F10' aufzurufen, wenn F10 gedrückt wird
hdlr.KEYBIND.add_call('Key_F10','on_keycall_F10')

# Hier patchen wir die ursprüngliche Handler-Datei, um eine neue Funktion hinzuzufügen,
# die unsere neue Funktion (mit demselben Namen) aufruft, definiert in dieser Datei.
self.on_keycall_F10 = types.MethodType(on_keycall_F10, self)

# Hier definieren wir eine Kopie der ursprünglichen Funktion 'on_keycall_F11'.
# damit wir sie später aufrufen können. Wir können jeden gültigen, unbenutzten
Funktionsnamen verwenden.
# Wir müssen dies tun, bevor wir die ursprüngliche Funktion überschreiben.
self.on_keycall_F11_super = self.on_keycall_F11

# Hier patchen wir die ursprüngliche Handler-Datei, um eine bestehende Funktion zu
überschreiben,
# und so auf unsere neue Funktion (mit demselben Namen) zu verweisen, definiert in dieser
Datei.
self.on_keycall_F11 = types.MethodType(on_keycall_F11, self)


# fügen sie einen neuen Pin dem Bildschirm hinzu:

# pin callback um den Status auszugeben
def new_pin_changed(data):
    print(data)

# Special function that gets called before the HAL component is set ready.
# Here we used the function to add a bool input pin with a callback.
def after_override__(self):
    try:
        pin = hdlr.QHAL.newpin("new_pin", hdlr.QHAL.HAL_BOOL, hdlr.QHAL.HAL_IN)
```

```
        pin.value_changed.connect(new_pin_changed)
    except Exception as e:
        print(e)

# Hier patchen wir die ursprüngliche Handler-Datei, um eine neue Funktion hinzuzufügen,
# die unsere neue Funktion (mit demselben Namen) aufruft, definiert in dieser Datei.
self.after_override__ = types.MethodType(after_override__, self)
```

Volle kreative Kontrolle mit benutzerdefinierten Handler/UI-Dateien

Wenn Sie einen Standardbildschirm mit voller Kontrolle **verändern** möchten, *kopieren Sie dessen UI und Handler-Datei in Ihren Konfigurationsordner*.

Es gibt ein QtVCP-Panel, das dabei hilft:

- Öffnen Sie ein Terminal und führen den folgenden Befehl aus:

  ```
  qtvcp copy
  ```

- Wählen Sie den Bildschirm und den Zielordner im Dialog
- Wenn Sie Ihren Bildschirm anders **benennen** möchten als den Standardnamen des eingebauten Bildschirms, ändern Sie den *Basisnamen* im Bearbeitungsfeld.
- Es sollte einen Ordner im Konfigurationsordner geben; für Bildschirme: mit dem Namen *<CONFIG FOLDER>/qtvcp/screens/* für Panels: mit dem Namen *<CONFIG FOLDER>/qtvcp/panels/* fügen Sie die Ordner hinzu, wenn sie fehlen, und kopieren Sie Ihre Ordner/Dateien hinein.
- Bestätigen, um alle Dateien zu kopieren
- Löschen Sie die Dateien, die Sie nicht ändern möchten, damit die Originaldateien verwendet werden.

12.6.3. VCP-Paneele

QtVCP kann verwendet werden, um Bedienfelder zu erstellen, die mit **HAL** verbunden sind.

Eingebaute Panels

Es sind mehrere **integrierte HAL-Panels** verfügbar.

Geben Sie in einem Terminal `qtvcp <return>` ein, um eine Liste zu sehen:

`test_panel`
: Sammlung nützlicher Widgets zum Testen von HAL-Komponenten, einschließlich der Anzeige des LED-Zustands.

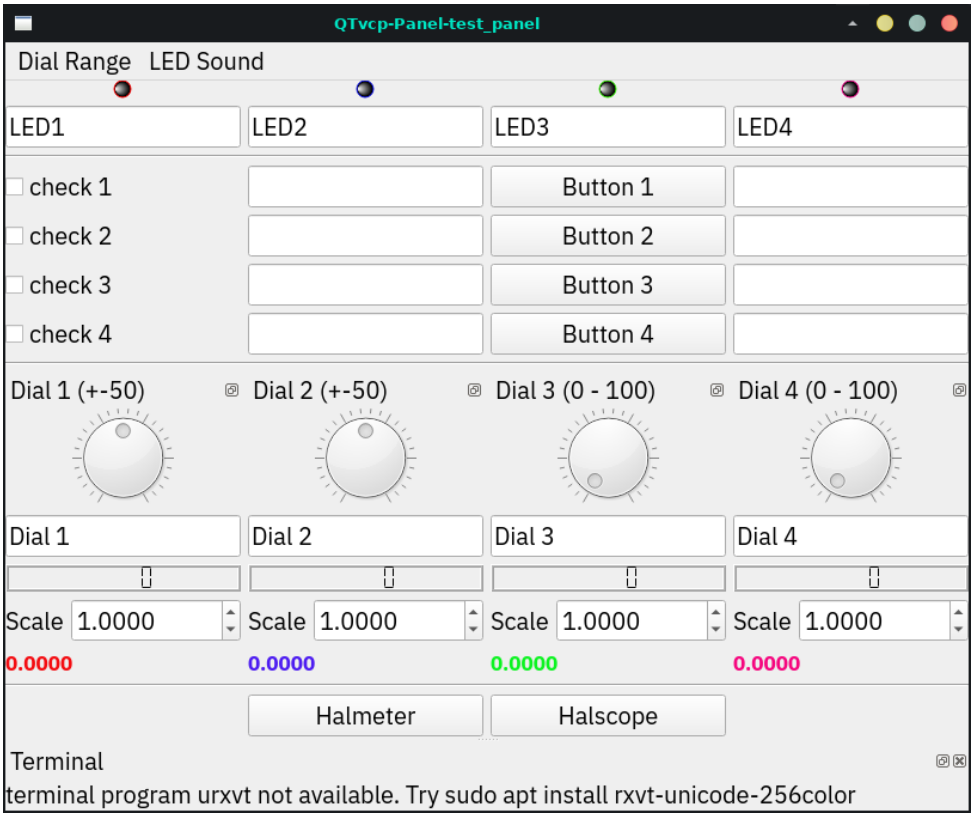


Figure 282. QtVCP HAL Test Integriertes Panel

cam_align

Ein Kameraanzeige-Widget für die Rotationsausrichtung.

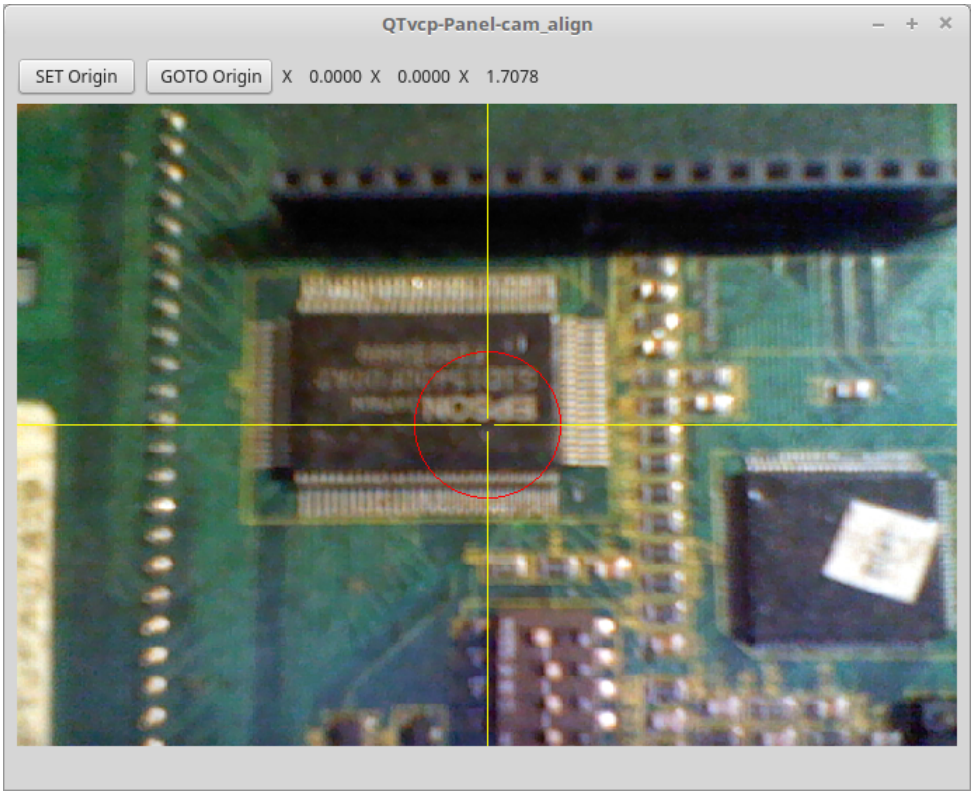


Figure 283. cam_align – Kameraausrichtung VCP

sim_panel

Ein kleines Bedienfeld zur Simulation von MPG-Jogging-Steuerungen usw.

Für simulierte Konfigurationen.



Figure 284. QtVCP Sim Eingebautes Panel

vismach_mill_xyz

3D-OpenGL-Ansicht einer *3-Achsen-Fräsmaschine*.

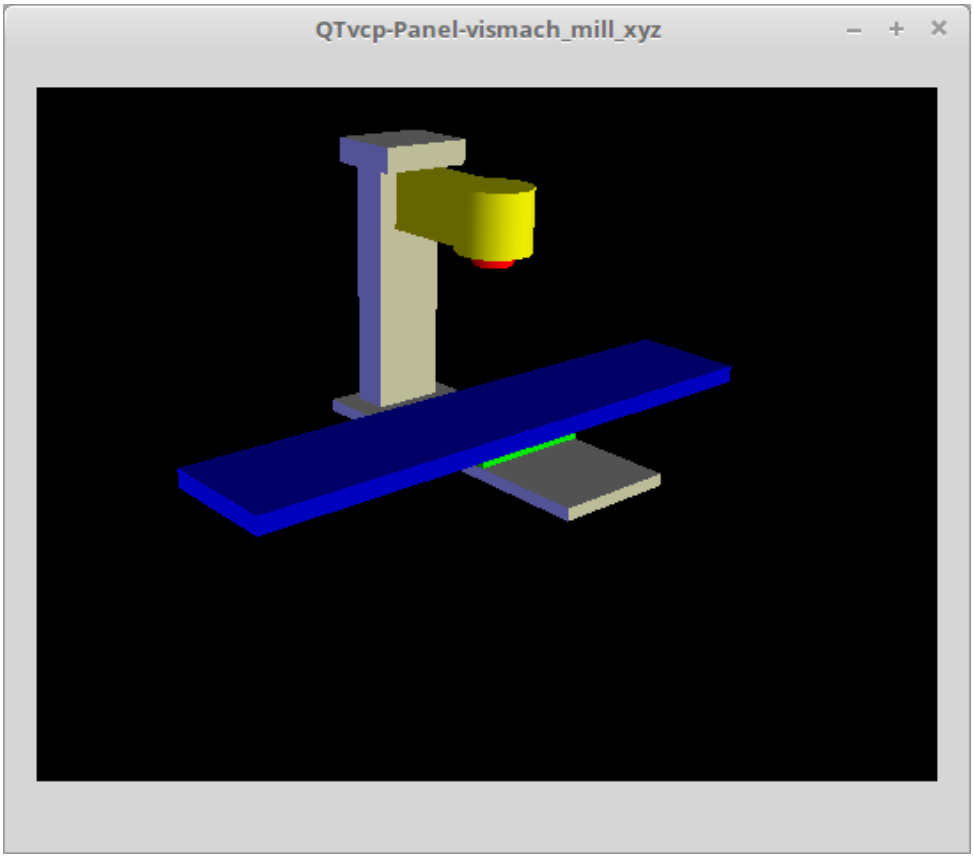


Figure 285. QtVismach - 3-Achsen-Fräse Eingebautes Panel

Sie können diese aus dem Terminal oder aus einer HAL-Datei mit diesem einfachen Befehl laden:

```
loadusr qtvcp test_panel
```

Aber typischerweise eher so:

```
loadusr -Wn test_panel qtvcp test_panel
```

Auf diese Weise wartet HAL bis die HAL-Pins gesetzt sind, bevor es weitergeht.

Benutzerdefinierte Bedienfelder

Sie können natürlich **Ihr eigenes Panel erstellen und laden**.

Wenn Sie eine UI-Datei mit dem Namen `my_panel.ui` und eine HAL-Datei mit dem Namen `my_panel.hal` erstellt haben, würden Sie diese dann von einem Terminal aus laden mit:

```
halrun -I -f my_panel.hal
```

Beispiel einer HAL-Datei, die ein QtVCP-Panel lädt

```
# Echtzeitkomponenten laden
loadrt threads
loadrt classicladder_rt

# Nicht-Echtzeit-Programme laden
```

```
loadusr classicladder
loadusr -Wn my_panel qtvcp my_panel.ui  ①

# Komponenten zum Thread hinzufügen
addf classicladder.0.refresh thread1


# connect pins
net bool-input1    test_panel.checkbox_1        classicladder.0.in-00
net bool-hide      test_panel.checkbox_4        classicladder.0.hide_gui

net bool-output1   test_panel.led_1             classicladder.0.out-00

net sint-in1       test_panel.doublescale_1-s   classicladder.0.s32in-00

# start thread
start
```

① In diesem Fall laden wir `qtvcp` mit **`-Wn`**, das wartet, bis das Panel das Laden beendet hat, bevor es mit der Ausführung des nächsten HAL-Befehls fortfährt.
Damit soll *gewährleistet werden, dass die vom Panel erstellten HAL-Pins tatsächlich fertig sind*, falls sie im Rest der Datei verwendet werden.

12.6.4. Erstellen eines einfachen benutzerdefinierten Bildschirms

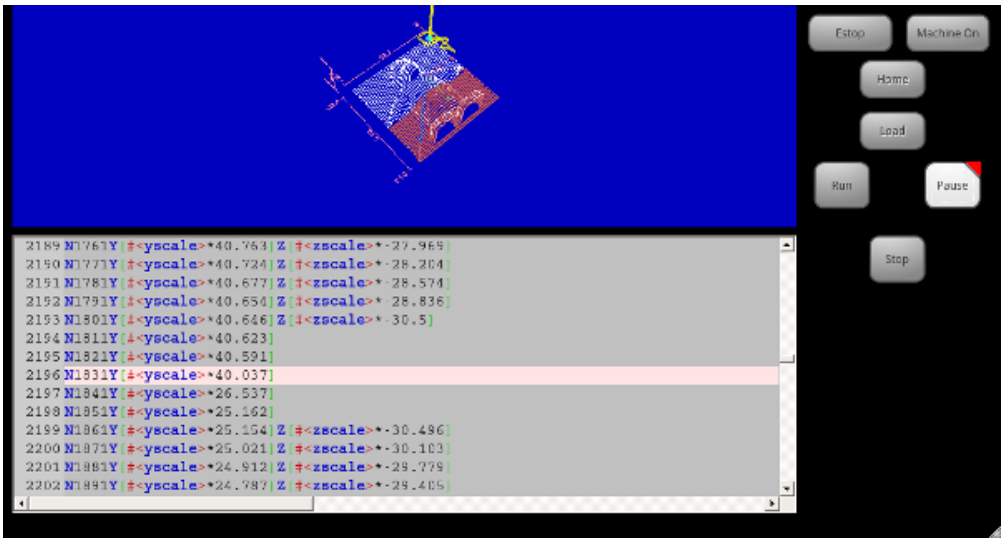


Figure 286. QtVCP Hässlicher benutzerdefinierter Bildschirm

Übersicht

So erstellen Sie ein Bedienfeld oder einen Bildschirm:

- Verwenden Sie Qt Designer, um ein Design zu erstellen, das Ihnen gefällt, und speichern Sie es in Ihrem Konfigurationsordner unter einem Namen Ihrer Wahl, der mit `.ui` endet
- Ändern Sie die Konfigurations-INI-Datei, um QtVCP mit Ihrer neuen `.ui`-Datei zu laden.
- Dann verbinden Sie alle erforderlichen HAL-Kenntnisse in einer HAL-Datei.

Holen Sie sich Qt Designer, um LinuxCNC-Widgets einzubinden

Qt Designer installieren

Zuerst müssen Sie den **Qt Designer installieren**.
Die folgenden Befehle sollten ihn zu Ihrem System hinzufügen, oder verwenden Sie Ihren Paketmanager, um dasselbe zu tun:

```
sudo apt-get install qttools5-dev-tools qttools5-dev libpython3-dev
```

Hinzufügen des Links `qtvcp_plugin.py` *zum Qt Designer Suchpfad*

Dann müssen Sie einen Link zu `qtvcp_plugin.py` in einem der Ordner hinzufügen, in denen Qt Designer suchen wird.

In einer *RIP* (engl. Abkürzung von "run in place", d.h. das Programm started dort wo es durch den Quellcode auch kompiliert wurde) Version von LinuxCNC wird `qtvcp_plugin.py` sein:

```
'~/LINUXCNC_PROJECT_NAME/lib/python/qtvcp/plugins/qtvcp_plugin.py'
```

Die *installierte Paketversion* sollte sein:

```
'usr/lib/python2.7/qtvcp/plugins/qtvcp_plugin.py' or
'usr/lib/python2.7/dist-packages/qtvcp/plugins/qtvcp_plugin.py'
```

Legen Sie einen symbolischen Link auf die obige Datei an und verschieben Sie sie an einen der Orte, an denen Qt Designer sucht.

Qt Designer sucht an diesen beiden Stellen nach Links (wählen Sie einen aus):

```
'/usr/lib/x86_64-linux-gnu/qt5/plugins/designer/python' or
'~/.designer/plugins/python'
```

Möglicherweise müssen Sie den Ordner `plugins/python` erstellen.

Starten Sie Qt Designer:

- Für eine *RIP-Installation*:
  Öffnen Sie ein Terminal, setzen Sie die Umgebungsvariablen für LinuxCNC <1>, dann laden Sie Qt Designer <2> mit :

  ```
  . scripts/rip-environment   ①
  designer -qt=5              ②
  ```

- Für eine *Paketinstallation*:
  Öffnen Sie ein Terminal und geben Sie ein:

  ```
  designer -qt=5
  ```

Wenn alles gut geht, wird Qt Designer gestartet und Sie werden die auswählbaren LinuxCNC Widgets

auf der linken Seite sehen.

Erstellen Sie die `.ui`-Datei des Bildschirms

Erstellen des `MainWindow` *Widgets*

Wenn Qt Designer zum ersten Mal gestartet wird, erscheint ein *'New Form' Dialog*.
Wählen Sie *'Main Window'* und drücken Sie die Schaltfläche *'Create'*.
Ein `*MainWindow*`*-Widget* wird angezeigt.

Wir werden diesem Fenster eine bestimmte, nicht veränderbare Größe geben:

Minimale und maximale Größe des Hauptfensters festlegen

- Fassen Sie die Ecke des Fensters an und ändern Sie die Größe auf eine geeignete Größe, z. B. 1000x600.
- Klicken Sie mit der rechten Maustaste auf das Fenster und klicken Sie auf *Mindestgröße* einstellen.
- Wiederholen Sie dies und stellen Sie *maximale Größe* ein.

Unser Beispiel-Widget ist nun nicht mehr größenveränderbar.

Hinzufügen des Widgets `ScreenOptions`

Ziehen Sie das `ScreenOptions`-Widget per Drag-and-Drop an eine beliebige Stelle im Hauptfenster.

Dieses Widget fügt visuell nichts hinzu, richtet aber einige **allgemeine Optionen** ein.

Es wird empfohlen, dieses Widget immer *vor allen anderen* hinzuzufügen.

Klicken Sie mit der rechten Maustaste auf das Hauptfenster, nicht auf das "ScreenOptions"-Widget, und stellen Sie "Layout" auf "Vertikal", um "ScreenOptions" in voller Größe anzuzeigen.

Panel-Inhalt hinzufügen

Auf der rechten Seite befindet sich ein Panel mit Registerkarten für einen *Eigenschaftseditor* und einen *Objektinspektor*.

Klicken Sie im Objektinspektor auf *ScreenOptions*.
Wechseln Sie dann zum Eigenschaftseditor (engl. property editor) und schalten Sie unter der Überschrift *ScreenOptions* die **`filedialog_option`** um.

Ziehen Sie ein **`GCodeGraphics`** *widget* und ein **`GcodeEditor`** *widget* per Drag and Drop.
Platzieren Sie sie und ändern Sie die Größe, wie Sie es für richtig halten, und lassen Sie etwas Platz für Schaltflächen.

Action Buttons hinzufügen

Fügen Sie dem Hauptfenster 7 Aktionsschaltflächen hinzu.

Wenn Sie auf die Schaltfläche (engl. button) doppelklicken, können Sie Text hinzufügen.
Bearbeiten Sie die Schaltflächenbeschriftungen für "Notaus" (engl. "E-stop"), "Maschine ein", Referenzpunkt (engl. "Home"), "Laden" (engl. "Load"), "Ausführen" (engl. "Run"), "Pause" und "Stopp".

Aktionsschaltflächen sind standardmäßig auf keine Aktion eingestellt, daher müssen wir die Eigenschaften für definierte Funktionen ändern. Sie können die Eigenschaften bearbeiten:

- direkt im *Eigenschaften-Editor* auf der rechten Seite des Qt-Designers, oder
- praktischerweise lassen sich durch einen Doppelklick mit der linken Maustaste auf die Schaltfläche ein Dialogfeld "Eigenschaften" aufrufen, was die Auswahl von Aktionen ermöglicht, wobei nur die für die Aktion relevanten Daten angezeigt werden.

Wir werden zunächst den bequemen Weg beschreiben:

- Klicken Sie mit der rechten Maustaste auf die Schaltfläche *Maschine ein* (engl.machine on) und wählen Sie *Aktionen festlegen* (engl. set actions).
- Wenn das Dialogfeld angezeigt wird, verwenden Sie die Combobox, um zu `MASCHINENSTEUERUNGEN - Maschine ein`(engl. `MACHINE CONTROLS - Machine On`) zu navigieren.
- In diesem Fall gibt es keine Option für diese Aktion, also wählen Sie "OK".

Jetzt schaltet die Taste das Gerät ein, wenn sie gedrückt wird.

Und nun der direkte Weg mit dem Eigenschaftseditor von Qt Designer:

- Wählen Sie die Schaltfläche "Maschine ein".
- Gehen Sie zum Eigenschaftseditor auf der rechten Seite von Qt Designer.
- Blättern Sie nach unten, bis Sie die Überschrift *ActionButton* finden.
- Klicken Sie auf das Kontrollkästchen für die Aktion "Machine_on", das Sie in der Liste der Eigenschaften und Werte sehen.

Die Taste steuert nun das Ein- und Ausschalten der Maschine.

Machen Sie das Gleiche für alle anderen Schaltflächen und fügen Sie noch einen hinzu:

- Bei der Schaltfläche "Home" müssen wir auch die Eigenschaft `joint_number` auf `1` ändern.
  Dadurch wird der Controller angewiesen, *alle Achsen* und nicht nur eine bestimmte Achse zu *referenzieren*.
- Mit dem Button "Pause":
  - Unter der Überschrift `Indicated_PushButton` überprüfen Sie die `Indicator_option`.
  - Unter der Überschrift `QAbstactButton` markieren Sie `checkable`.

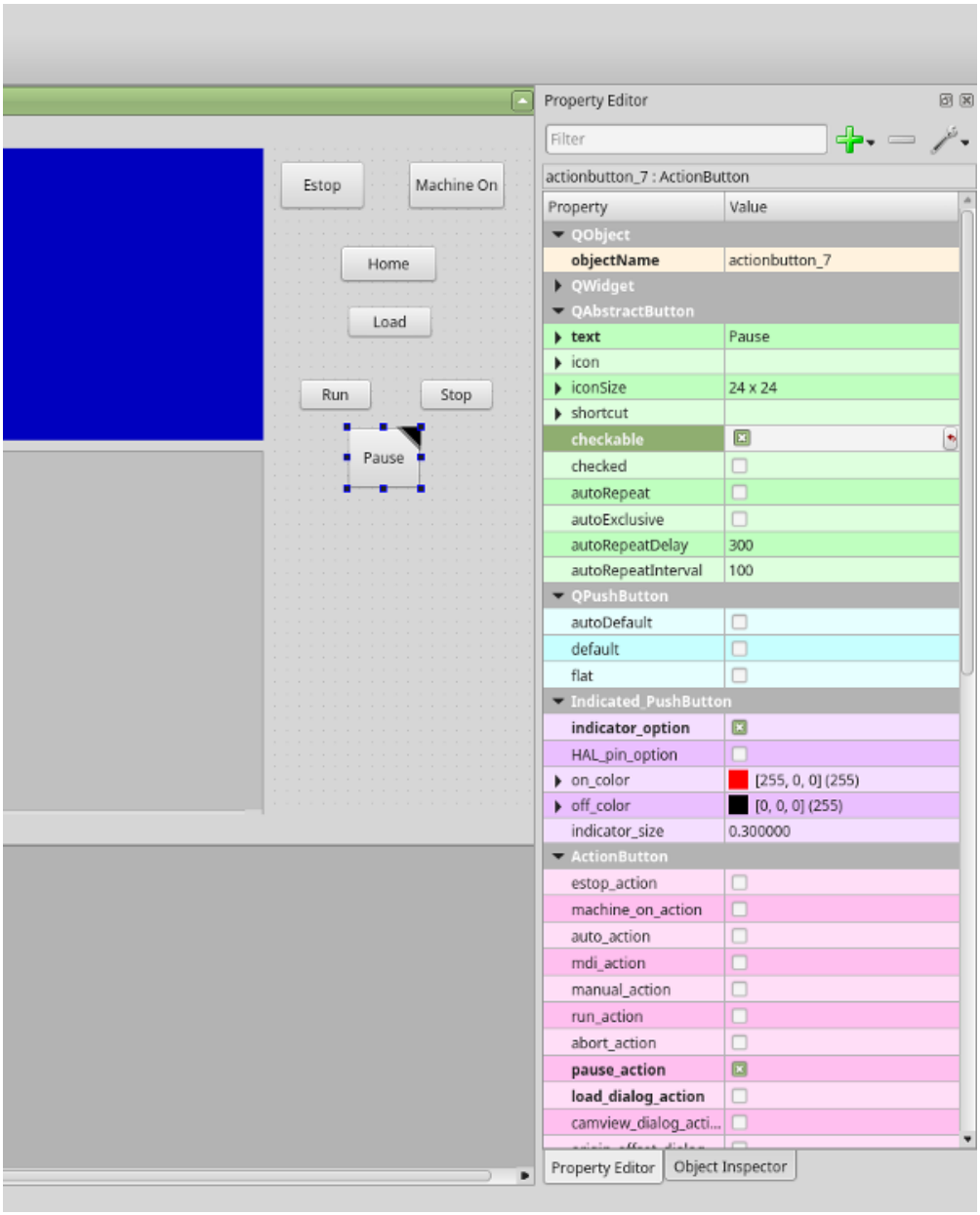


Figure 287. Qt Designer: Auswahl der Eigenschaften des Pause-Buttons (Schaltfläche)

`.ui`-*Datei speichern*

Diesen Entwurf müssen wir dann als `tester.ui` im Ordner `sim/qtvcp` speichern.

Wir speichern sie unter dem Namen *tester*, da dies ein Dateiname ist, den QtVCP erkennt und eine eingebaute Handler-Datei verwendet, um sie anzuzeigen.

Handler-Datei

Eine Handler-Datei ist **erforderlich**.

Er ermöglicht das Schreiben von Anpassungen in Python.

Zum Beispiel werden *Tastatursteuerungen* normalerweise in die Handler-Datei geschrieben.

In diesem Beispiel wird die eingebaute Datei `tester_handler.py` automatisch verwendet: Sie tut das Minimum, das erforderlich ist, um den in `tester.ui` definierten Bildschirm darzustellen und einfache

Tastatureingaben vorzunehmen.

INI-Konfiguration

[DISPLAY] Abschnitt

Wenn Sie QtVCP zur Erstellung eines CNC-Steuerungsbildschirms verwenden, setzen Sie unter der Überschrift *INI-Datei* `[DISPLAY]`:

```
DISPLAY = qtvcp <Bildschirmname>
```

`_<Bildschirmname>_` ist der *Basisname* der Dateien `.ui` und `_handler.py`.

In unserem Beispiel gibt es bereits eine Sim-Konfiguration namens tester, die wir zur Anzeige unseres Testbildschirms verwenden werden.

[HAL] Abschnitt

Wenn Ihr Bildschirm *Widgets mit HAL-Pins* verwendet, dann müssen Sie diese **in einer HAL-Datei** verbinden.

QtVCP sucht in der *INI-Datei* unter der Überschrift `[HAL]` nach den folgenden Einträgen:

`POSTGUI_HALFILE=<Dateiname>`

Der Konvention nach wäre `<Dateiname>` als `+<Bildschirm_name>_postgui.hal+` genannt, aber es kann jeder legale Dateiname sein.
Sie können *mehrere `POSTGUI_HALFILE`-Zeilen* in der INI haben: jede wird nacheinander in der Reihenfolge ausgeführt, in der sie erscheint.
Diese Befehle werden *nach der Erstellung des Bildschirms ausgeführt*, um sicherzustellen, dass die HAL-Pins des Widgets verfügbar sind.

`POSTGUI_HALCMD=<Befehl>`

`<Befehl>` wäre *jeder gültige HAL-Befehl.*
Sie können *mehrere `POSTGUI_HALCMD`-Zeilen* in der INI haben: jede wird nacheinander in der Reihenfolge ausgeführt, in der sie erscheint.
Um zu garantieren, dass die HAL-Pins des Widgets verfügbar sind, werden diese Befehle ausgeführt:

- *nachdem der Bildschirm gebaut ist,*
- *nachdem alle POSTGUI_HALFILEs ausgeführt wurden.*

In unserem Beispiel gibt es keine HAL-Pins zu verbinden.

12.6.5. Handler-Datei im Detail

Handler-Dateien werden zur *Erstellung von benutzerdefinierten Steuerelementen mit Python* verwendet.

Übersicht

Hier ist ein Beispiel für eine Handler-Datei.

Es ist in Abschnitte unterteilt, um die Diskussion zu erleichtern.

```
############################
# **** IMPORT SECTION **** #
############################
import sys
import os
import linuxcnc

from PyQt5 import QtCore, QtWidgets

from qtvcp.widgets.mdi_line import MDILine as MDI_WIDGET
from qtvcp.widgets.gcode_editor import GcodeEditor as GCODE
from qtvcp.lib.keybindings import Keylookup
from qtvcp.core import Status, Action

# Set up logging
from qtvcp import logger
LOG = logger.getLogger(__name__)

# Set the log level for this module
#LOG.setLevel(logger.INFO) # One of DEBUG, INFO, WARNING, ERROR, CRITICAL

############################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
############################################

KEYBIND = Keylookup()
STATUS = Status()
ACTION = Action()
###################################
# **** HANDLER CLASS SECTION **** #
###################################

class HandlerClass:

    ########################
    # **** INITIALIZE **** #
    ########################
    # Widgets ermöglicht den Zugriff auf Widgets aus den QtVCP-Dateien
    # An dieser Stelle werden die Widgets und HAL-Pins nicht instanziiert
    def __init__(self, halcomp,widgets,paths):
        self.hal = halcomp
        self.w = widgets
        self.PATHS = paths

    ########################################
    # SPECIAL FUNCTIONS SECTION            #
    ########################################

    # at this point:
    # the widgets are instantiated.
    # the HAL pins are built but HAL is not set ready
    # This is where you make HAL pins or initialize state of widgets etc
    def initialized__(self):
        pass
```

```
    def processed_key_event__(self,receiver,event,is_pressed,key,code,shift,cntrl):
        # when typing in MDI, we don't want keybinding to call functions
        # so we catch and process the events directly.
        # We do want ESC, F1 and F2 to call keybinding functions though
        if code not in(QtCore.Qt.Key_Escape,QtCore.Qt.Key_F1 ,QtCore.Qt.Key_F2,
                    QtCore.Qt.Key_F3,QtCore.Qt.Key_F5,QtCore.Qt.Key_F5):

            # search for the top widget of whatever widget received the event
            # then check if it is one we want the keypress events to go to
            flag = False
            receiver2 = receiver
            while receiver2 is not None and not flag:
                if isinstance(receiver2, QtWidgets.QDialog):
                    flag = True
                    break
                if isinstance(receiver2, MDI_WIDGET):
                    flag = True
                    break
                if isinstance(receiver2, GCODE):
                    flag = True
                    break
                receiver2 = receiver2.parent()

            if flag:
                if isinstance(receiver2, GCODE):
                    # if in manual do our keybindings - otherwise
                    # send events to G-code widget
                    if STATUS.is_man_mode() == False:
                        if is_pressed:
                            receiver.keyPressEvent(event)
                            event.accept()
                        return True
                elif is_pressed:
                    receiver.keyPressEvent(event)
                    event.accept()
                    return True
                else:
                    event.accept()
                    return True

        if event.isAutoRepeat():return True

        # ok if we got here then try keybindings
        try:
            return KEYBIND.call(self,event,is_pressed,shift,cntrl)
        except NameError as e:
            LOG.debug('Exception in KEYBINDING: {}'.format (e))
        except Exception as e:
            LOG.debug('Exception in KEYBINDING:', exc_info=e)
            print('Error in, or no function for: %s in handler file for-%s'%(KEYBIND
.convert(event),key))
            return False

    ########################
    # CALLBACKS FROM STATUS #
```

```
    ########################

    ######################
    # CALLBACKS FROM FORM #
    ######################

    ####################
    # GENERAL FUNCTIONS #
    ####################

    # keyboard jogging from key binding calls
    # double the rate if fast is true
    def kb_jog(self, state, joint, direction, fast = False, linear = True):
        if not STATUS.is_man_mode() or not STATUS.machine_is_on():
            return
        if linear:
            distance = STATUS.get_jog_increment()
            rate = STATUS.get_jograte()/60
        else:
            distance = STATUS.get_jog_increment_angular()
            rate = STATUS.get_jograte_angular()/60
        if state:
            if fast:
                rate = rate * 2
            ACTION.JOG(joint, direction, rate, distance)
        else:
            ACTION.JOG(joint, 0, 0, 0)

    ####################
    # KEY BINDING CALLS #
    ####################

    # Machine control
    def on_keycall_ESTOP(self,event,state,shift,cntrl):
        if state:
            ACTION.SET_ESTOP_STATE(STATUS.estop_is_clear())
    def on_keycall_POWER(self,event,state,shift,cntrl):
        if state:
            ACTION.SET_MACHINE_STATE(not STATUS.machine_is_on())
    def on_keycall_HOME(self,event,state,shift,cntrl):
        if state:
            if STATUS.is_all_homed():
                ACTION.SET_MACHINE_UNHOMED(-1)
            else:
                ACTION.SET_MACHINE_HOMING(-1)
    def on_keycall_ABORT(self,event,state,shift,cntrl):
        if state:
            if STATUS.stat.interp_state == linuxcnc.INTERP_IDLE:
                self.w.close()
            else:
                self.cmnd.abort()

    # Linear Jogging
    def on_keycall_XPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 0, 1, shift)
```

```
    def on_keycall_XNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 0, -1, shift)

    def on_keycall_YPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 1, 1, shift)

    def on_keycall_YNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 1, -1, shift)

    def on_keycall_ZPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 2, 1, shift)

    def on_keycall_ZNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 2, -1, shift)

    def on_keycall_APOS(self,event,state,shift,cntrl):
        pass
        #self.kb_jog(state, 3, 1, shift, False)

    def on_keycall_ANEG(self,event,state,shift,cntrl):
        pass
        #self.kb_jog(state, 3, -1, shift, linear=False)

    ############################
    # **** closing event **** #
    ############################

    ##############################
    # required class boiler code #
    ##############################

    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)

################################
# required handler boiler code #
################################

def get_handlers(halcomp,widgets,paths):
     return [HandlerClass(halcomp,widgets,paths)]
```

IMPORT Bereich

Dieser Abschnitt ist für **Import der erforderlichen Bibliotheksmodule** für Ihren Bildschirm.

Es wäre typisch, die QtVCP-Bibliotheken *keybinding*, *Status* und *Action* zu importieren.

Abschnitt INSTANTIATE BIBRARIES

Indem wir die Bibliotheken hier instanziieren, **erzeugen wir eine globale Referenz**.

Sie können dies an den Befehlen erkennen, denen kein "self" vorangestellt ist.

Konventionell werden die Namen von global referenzierten Bibliotheken *großgeschrieben*.

HANDLER CLASS-Abschnitt

Der **angepasste Code** wird *in einer Klasse platziert, damit QtVCP ihn verwenden kann*.

Dies ist die Definition der Handler-Klasse.

INITIALIZE Abschnitt

Wie alle Python-Bibliotheken wird die **`+__init__+`-Funktion** aufgerufen, wenn die Bibliothek *erstmals* instanziiert wird.

Hier können Sie *Standardwerte*, *Referenzvariablen* und *globale Variablen* einrichten.

Die Referenzen der Widgets sind zu diesem Zeitpunkt nicht verfügbar.

Die Variablen `halcomp`, `widgets` und `paths` ermöglichen den Zugriff auf QtVCP's HAL-Komponenten, Widgets bzw. Pfadinformationen.

Abschnitt zu BESONDEREN FUNKTIONEN

Es gibt mehrere *spezielle Funktionen*, nach denen QtVCP in der Handler-Datei sucht. Wenn QtVCP diese findet, ruft es sie auf, wenn nicht, ignoriert es sie stillschweigend.

`class_patch__(self):`

Class patching (Klassen-Patching), auch bekannt als **monkey patching**, erlaubt es, **Funktionsaufrufe in einem importierten Modul zu überschreiben**.
Class patching muss *vor der Instanziierung des Moduls* erfolgen und *ändert alle Instanzen*, die danach erstellt werden.
Ein Beispiel wäre das Patchen von Button-Aufrufen aus dem G-Code-Editor, sodass stattdessen Funktionen in der Handler-Datei aufgerufen werden.
Hier neu definierte Class-Patching-Funktionen werden mit der HandlerClass-Instanz als *self* aufgerufen, nicht mit der Instanz der gepatchten Klasse. Dies kann den Zugriff auf Funktionen oder Variablen der gepatchten Klasse erschweren.
Beim Class Patching außerhalb der HandlerClass-Klasse verwendet der Funktionsaufruf die Instanz der gepatchten Klasse als *self*.

`initialized__(self):`

Diese Funktion wird *aufgerufen, nachdem die Widgets und HAL-Pins erstellt wurden*.
Sie können hier die Widgets und HAL-Pins manipulieren oder weitere HAL-Pins hinzufügen.
In der Regel gibt es

- Einstellungen überprüft und eingestellt,
- auf Widgets angewendete Stile,
- Status von LinuxCNC verbunden mit Funktionen.
- Tastenbelegungen würden hinzugefügt.

pre_hal_init__(self):

Diese Funktion wird aufgerufen, bevor die HAL-integrierten Widgets ihre hal_init_-Funktion aufrufen.
Einige Eigenschaften müssen geändert werden, bevor HAL_init für das Widget aufgerufen wird.

after_override__(self):

Diese Funktion wird aufgerufen, nachdem die optionale Override-Datei geladen ist,
aber bevor die optionale HAL-Datei geladen oder HAL-Komponente bereit ist.

processed_key_event__(self,receiver,event,is_pressed,key,code,shift,cntrl):

Diese Funktion wird aufgerufen, um *Tastatur-Jogging* usw. zu erleichtern.
Durch die Verwendung der *`keybinding`-Bibliothek* kann dies verwendet werden, um einfach Funktionen hinzuzufügen, die an Tastendrücke gebunden sind.

keypress_event__(self,receiver, event):

Diese Funktion liefert **rohe Tastendruckereignisse**.
Sie hat _Vorrang vor dem `verarbeiteten_Tastenereignis`.

keyrelease_event__(receiver, event):

Diese Funktion gibt **raw key release events** aus.
Es hat *Vorrang vor* dem `processed_key_event`.

before_loop__(self):

Diese Funktion wird *kurz vor dem Eintritt in die Qt-Ereignisschleife* aufgerufen. Zu diesem Zeitpunkt sind alle Widgets/Bibliotheken/Initialisierungscodes abgeschlossen und der Bildschirm wird bereits angezeigt.

system_shutdown_request__(self):

Falls vorhanden, überschreibt diese Funktion **die normale Funktion, die beim vollständigen Herunterfahren des Systems aufgerufen wird**.
Sie kann dazu benutzt werden, *vor dem Herunterfahren Hausarbeiten* zu erledigen.
+
Das Linux *System wird nicht heruntergefahren, wenn Sie diese Funktion verwenden*, Sie müssen das selbst tun.
QtVCP/LinuxCNC beendet sich ohne eine Eingabeaufforderung, sobald diese Funktion zurückkehrt.

closing_cleanup__(self):

Diese Funktion wird *kurz vor dem Schließen des Bildschirms* aufgerufen. Sie kann verwendet werden, um vor dem Schließen aufzuräumen.

STATUS CALLBACKS Abschnitt

Konventionell würden Sie hier Funktionen unterbringen, die **Rückrufe von STATUS-Definitionen** sind.

CALLBACKS FROM FORM Abschnitt

Konventionell würden Sie hier Funktionen ablegen, die **Rückrufe von den Widgets sind, die mit dem**

MainWindow im Qt Designer-Editor verbunden sind.

Abschnitt mit ALLGEMEINEN FUNKTIONEN

Konventionell werden hier die **allgemeinen Funktionen** untergebracht.

Abschnitt zur KEY BINDING (engl. für Tastenbelegung)

Wenn Sie die `Keybinding`-Bibliothek_ verwenden, platzieren Sie hier Ihre **benutzerdefinierten Tastenaufrufroutinen**.

Die Funktionssignatur ist:

```
def on_keycall_KEY(self,event,state,shift,cntrl):
    if state:
        self.do_something_function()
```

`KEY` ist der Code (aus der Keybindings-Bibliothek) für den gewünschten Schlüssel.

CLOSING EVENT Sektion

Wenn Sie die Funktion **`closeEvent` hier einfügen, werden Schließungsereignisse** abgefangen.

Dies *replaces eine vordefinierte 'closeEvent'*-Funktion von QtVCP.

```
def closeEvent(self, event):
    self.do_something()
    event.accept()
```

NOTE Normalerweise ist es besser, die spezielle Funktion `closing_cleanup__` zu verwenden.

12.6.6. Verbinden von Widgets mit Python-Code

Es ist möglich, Widgets über **Signale und Slots** mit Python-Code zu verbinden.

Auf diese Weise können Sie:

- *LinuxCNC-Widgets neue Funktionen geben*, oder
- *Standard Qt-Widgets zur Steuerung von LinuxCNC verwenden*.

Übersicht

Im Qt Designer-Editor:

- Sie *erstellen Benutzerfunktions-Slots*
- Sie *verbinden die Slots mit Widgets, indem Sie Signale verwenden*.

In der Handler-Datei:

- Sie *erstellen die Funktionen des Slots*, die im Qt Designer definiert sind.

Hinzufügen von Slots mit Qt Designer

Wenn Sie Ihren Bildschirm in Qt Designer geladen haben, fügen Sie einen einfachen `PushButton` zu dem Bildschirm hinzu.
Sie könnten den Namen der Schaltfläche in etwas Interessantes wie "test_button" ändern.

Es gibt zwei Möglichkeiten, Verbindungen zu bearbeiten - Dies ist die grafische Methode.

- In der oberen Werkzeugleiste von Qt Designer gibt es eine Schaltfläche zum Bearbeiten von Signalen. Wenn Sie auf die Schaltfläche klicken und sie gedrückt halten, wird ein Pfeil angezeigt (sieht aus wie ein Erdungssignal aus einem elektrischen Schaltplan).
- Schieben Sie diesen Pfeil auf einen Bereich des Hauptfensters, in dem sich keine Widgets befinden.
- Ein Dialogfeld „Verbindungen konfigurieren“ wird angezeigt.
  - Die Liste auf der linken Seite enthält die verfügbaren Signale des Widgets.
  - Die Liste auf der rechten Seite sind die verfügbaren Slots im Hauptfenster und Sie können sie ergänzen.
- Wählen Sie das Signal `clicked()` - dies macht die Slot-Seite verfügbar.
- Klicken Sie in der Slotliste auf "Bearbeiten" (engl. edit).
- Ein Dialogfeld "Slots/Signale des Hauptfensters" wird angezeigt.
- In der Slots-Liste oben befindet sich ein "+"-Symbol - klicken Sie darauf.
- Sie können nun einen neuen Slotnamen bearbeiten.
- Löschen Sie den Standardnamen `slot()` und ändern Sie ihn in `test_button()`.
- Drücken Sie die Taste *OK*.
- Sie gelangen zurück zum Dialog *Verbindungen konfigurieren*.
- Nun können Sie Ihren neuen Slot in der Slotliste auswählen.
- Drücken Sie dann auf "OK" und speichern Sie die Datei.

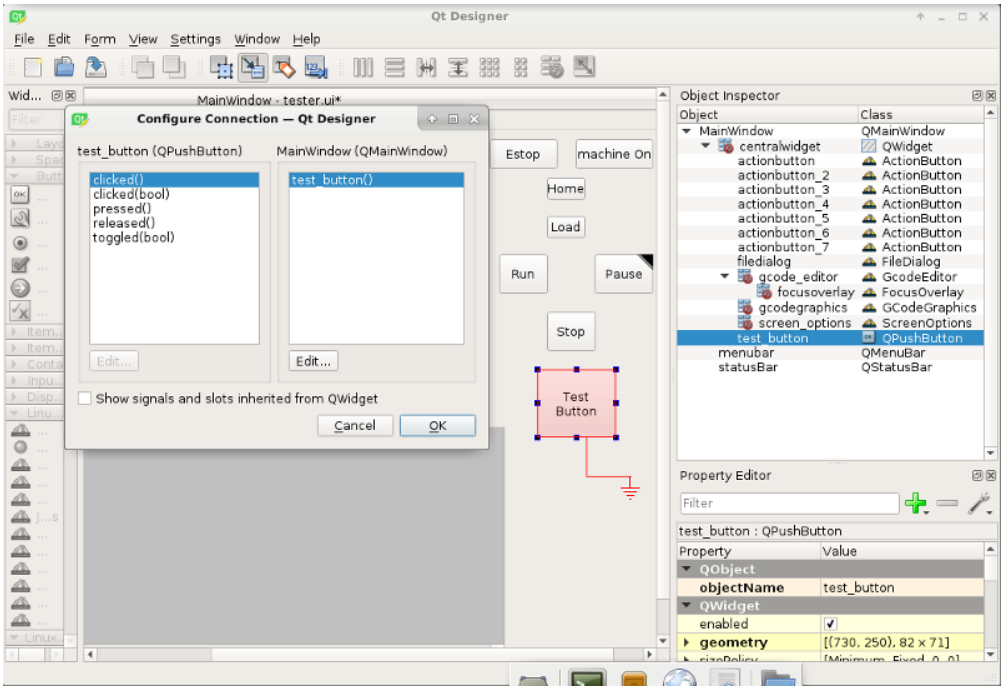


Figure 288. Qt Designer-Signal/Slot-Auswahl

Änderungen am Python-Handler

Nun müssen Sie **die Funktion in die Handler-Datei einfügen**.

Die Funktionssignatur lautet `def slot_name(self):`.

Für unser Beispiel fügen wir etwas Code hinzu, um den Namen des Widgets zu auszugeben:

```
def test_button(self):
    name = self.w.sender().text()
    print(name)
```

Fügen Sie diesen Code unter dem Abschnitt namens:

```
########################
# Callbacks vom Formular
########################
```

Tatsächlich spielt es keine Rolle, wo in der Handler-Klasse Sie die Befehle ablegen, aber per Konvention ist dies der Ort, an dem Sie sie ablegen müssen.

Speichern der Handlerdatei.

Wenn Sie nun Ihren Bildschirm laden und die Schaltfläche drücken, sollte der Name der Schaltfläche im Terminal angezeigt werden.

12.6.7. Mehr zum Thema

QtVCP Builtin Virtual Control Panels

QtVCP Widgets

QtVCP Libraries

Qt Vismach

QtVCP Handler Datei Code Kurz-Beispiele

QtVCP Entwicklung

QtVCP Angepasste Qt Designer Widgets

12.7. QtVCP Virtuelle Kontrollpanels

QtVCP kann zur **Erstellung von Bedienfeldern** verwendet werden, die mit *HAL* verbunden sind.

12.7.1. Eingebaute virtuelle Bedienpulte (engl. virtual control panels)

Es sind mehrere **integrierte HAL-Panels** verfügbar.

Geben Sie in einem Terminal `qtvcp list` ein, um eine Liste zu sehen.

`copy` (engl. für kopieren)

Dient zum **Kopieren von QtVCP's eingebautem Screens/VCP Panels/QtVismach Code in einen Ordner**, damit man ihn *anpassen* kann.

In einem Terminal ausführen:

```
qtvcp copy
```

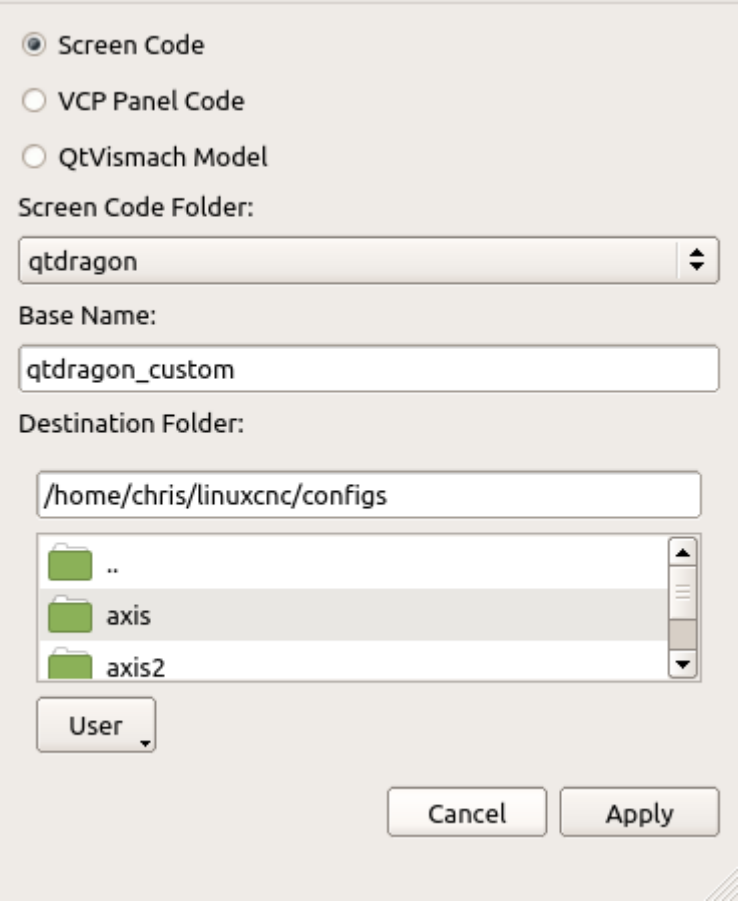
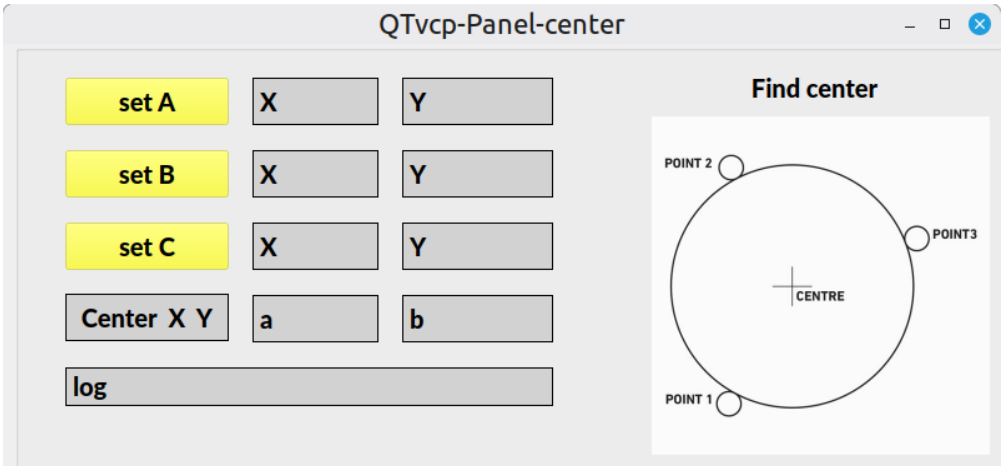


Figure 289. QtVCP `copy` Dialog - Bildschirm, VCP Panel oder QtVismach Code Kopieren Panel

`center`



Used for **finding center of three points**.

In einem Terminal ausführen:

```
qtvcp center
```

Qt Embedding Commands

Fügen Sie diese Zeilen in den Abschnitt [DISPLAY] in Ihrer .ini-Datei ein

Das Beispiel tab_location ist für den QtDragon-Bildschirm.

```
EMBED_TAB_NAME=Center Finder
EMBED_TAB_COMMAND=qtvcp center
EMBED_TAB_LOCATION=tabWidget_utilities
```

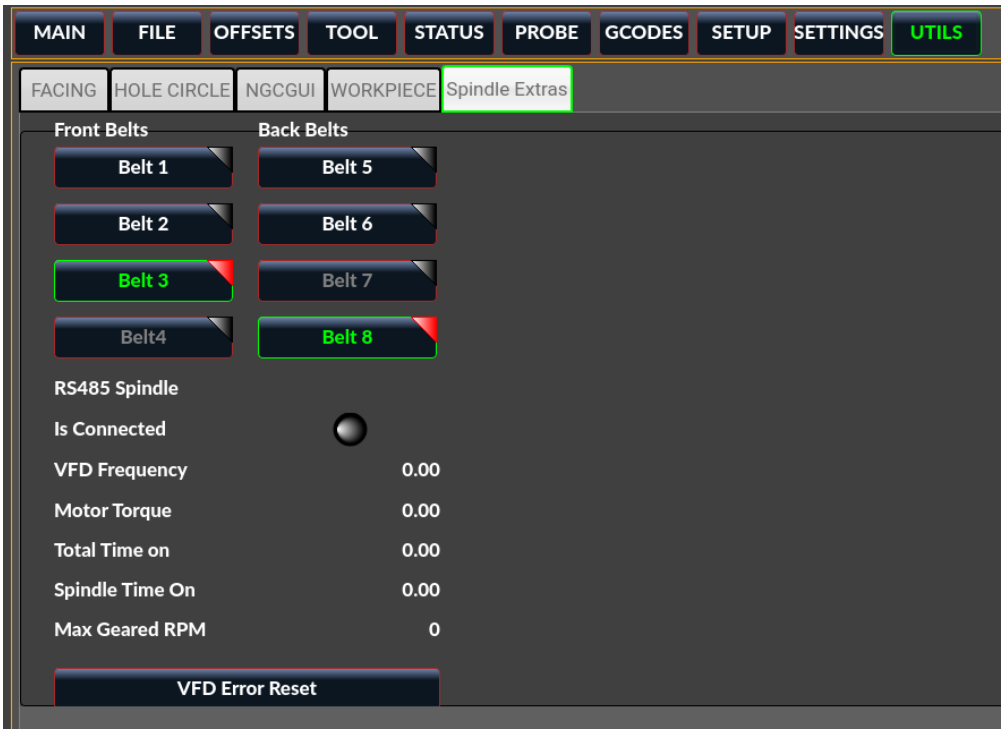
AXIS Embedding Commands

Add these lines to the [DISPLAY] section in your .ini file

```
# Embed in tabs
EMBED_TAB_NAME= Center Finder
EMBED_TAB_COMMAND= halcmd loadusr -Wn qt_embed qtvcp -d -c qt_embed  -x {XID} center
```

spindle_belts

Dieses Panel dient zur Anzeige zusätzlicher RS485-VFD-Daten und zur Konfiguration eines Spindelantriebs mit 4 Scheiben und 2 Riemen über eine Reihe von Tasten.



Darüber hinaus ist es auch eine nützliche Vorlage für Ihr benutzerdefiniertes Panel, denn es enthält:

- Anzeige von zusätzlichen HAL Daten
- Schaltflächen (engl. buttons) und Schaltflächengruppen (engl. button groups)
- Dynamische Änderungen des Aktivierungs-/Deaktivierungsstatus von Schaltflächen (engl. buttons) basierend auf dem Status anderer Schaltflächen
- Speichern von Daten in die qtdragon.prefs Datei
- Benutzerdefinierte Taste zum Zurücksetzen des VFD

Passen Sie dieses Panel an Ihre eigenen Anforderungen an. Die meisten gängigen Funktionen werden verwendet. Der Vorteil der Verwendung von Panels besteht darin, dass Ihr benutzerdefinierter Anzeigecode vom qtdragon-Kerncode getrennt wird, so dass ein Upgrade des Systems Ihre Anpassungen nicht zerstört.

Zusätzliche Anforderungen

- Einen Spindelantrieb (z.B. VFDMOD)
- Eine benutzerdefinierte Komponente zur Skalierung der Frequenz des VFD, um die gewünschte Spindeldrehzahl zu erhalten.
- Eine riemengetriebene Spindel mit zwei Riemen und einer dazwischenliegenden Umlenkrolle, ähnlich wie bei einer Bohrmaschine.
- Verbinden der Eingangspins qtdragon.belts.<pin-name> in Ihrer postgui HAL Datei.

Funktion

Die Bänder werden in zwei Gruppen von Buttons aufgeteilt, die Vordergurte (engl. front belts) und die Hintergurte (engl. rear belts) zerlegt. Diese sind je nach Platte auf der Maschine nummeriert. Buttons in einer Gruppe sind einander ausschließend, d.h. nur einer kann in der Gruppe ausgewählt werden.

Außerdem ist es bei dieser Art von Mechanismus nicht möglich, beide Riemen auf der gleichen Ebene zu haben, da man nicht zwei Riemen auf eine Umlenkrolle auflegen kann. Wenn also ein Riemen ausgewählt ist, ist die gegenüberliegende Taste deaktiviert. Wenn z. B. Riemen 3 ausgewählt ist, ist Riemen 7 deaktiviert.

Befehle einbetten

Fügen Sie diese Zeilen in den Abschnitt [DISPLAY] in Ihrer .ini-Datei ein
Das Beispiel tab_location ist für den QtDragon-Bildschirm.

```
EMBED_TAB_NAME=Spindle Extras
EMBED_TAB_COMMAND=qtvcp spindle_belts
EMBED_TAB_LOCATION=tabWidget_utilities
```

So laden Sie `spindle_belt ` aus einem HAL-Skript:

```
loadusr qtvcp spindle_belts
```

Hinweise zur Anpassung

Anpassen des Panels:

- Kopieren Sie die Dateien in /user/share/qtvcp/qtdragon/panels/belts nach: ~/linuxcnc/configs/<Mein_Konfigurationsordner>/qtvcp/panels/belts (Sie können dazu das Kopierdialogfeld verwenden)
- Bearbeiten Sie belts.ui mit Designer.

- Bearbeiten Sie Belts_handler.py mit einem Texteditor
- Verbinden Sie die relevanten Pins in der Datei postgui.hal
- Stellen Sie sicher, dass Ihre Postgui-Datei von Ihrer .ini-Datei geladen wird.

Informationen zu den Feinheiten finden Sie in der QtVCP- und QtDragon-Dokumentation. Die Python-Handler-Datei bietet auch eine nützliche Vorlage für jedes benutzerdefinierte Panel.

`test_dial`

- This panel has a **dial that adjusts SINT and REAL HAL output pins**.
- Der Bereich des Zifferblatts kann über ein Dropdown-Menü eingestellt werden.
- Die Ausgabe kann mit der `spinbox` skaliert werden.
- Eine `combobox` kann verwendet werden, um automatisch ein Signal auszuwählen und eine Verbindung herzustellen.

```
loadusr qtvcp test_dial
```

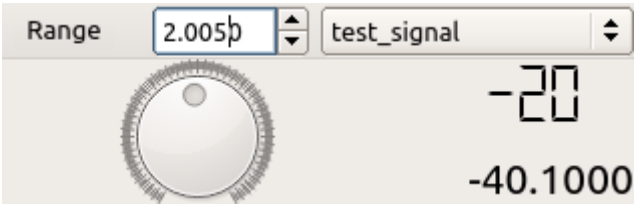


Figure 290. QtVCP `test_dial` Panel - Testwahl VCP

test_button

- Dieses Panel hat eine **Schaltfläche, die einen HAL-Pin** setzt.
- Die Schaltfläche kann als *Momentan-* oder als *Umschalt*-Schaltfläche ausgewählt werden.
- Ein HAL-Pin wird erstellt, der dem Zustand des Buttons folgt.
- Die *Indikatorfarbe* der Schaltfläche kann über ein Dropdown-Menü eingestellt werden.
- Ein HAL-Pin oder Signal kann ausgewählt werden, um dem Zustand des Buttons zu folgen.
- Sie können weitere Schaltflächen über das Dropdown-Menü hinzufügen.
- Sie können ein Halmeter aus dem Dropdown-Menü laden.
- Sie können eine Test-LED aus dem Dropdown-Menü laden.
- Die Schaltfläche kann von den Hauptfenstern abgetrennt werden.

So laden Sie `test_button` aus einem HAL-Skript:

```
loadusr qtvcp test_button
loadusr qtvcp -o 4 test_button
```

Der `-o` Schalter legt fest, mit wie vielen Knöpfen das Panel startet.
Wenn Sie direkt von einem Terminal laden, lassen Sie das `loadusr` weg.

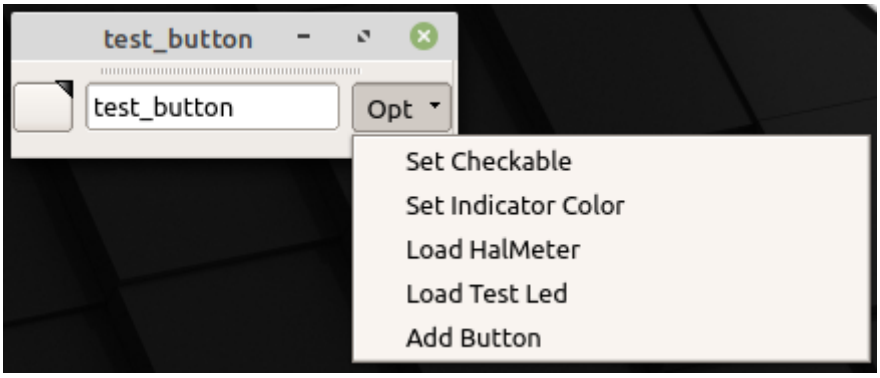


Figure 291. QtVCP `test_button` - Test Button VCP

`test_led`

- Dieses Panel verfügt über eine *LED, die ausgewählt werden kann, um HAL-Bit-Pins / Signale * zu beobachten.
- Die Farbe der LED kann über ein Dropdown-Menü eingestellt werden.
- Das Textfeld und der Status können als Sprache ausgegeben werden, wenn Ton ausgewählt ist.
- Eine `combobox` kann zur automatischen Auswahl und Verbindung mit einem Pin/Signal verwendet werden.
- Sie können weitere LEDs über das Dropdown-Menü hinzufügen.
- Die LED kann von den Hauptfenstern abgenommen werden.

So laden Sie `test_led` aus einem HAL-Skript:

```
loadusr qtvcp test_led
loadusr qtvcp -o 4 test_led
```

Der *-o*-Schalter stellt ein, mit wie vielen LEDs das Panel startet.
Wenn Sie direkt von einem Terminal laden, lassen Sie den *loadusr* weg.

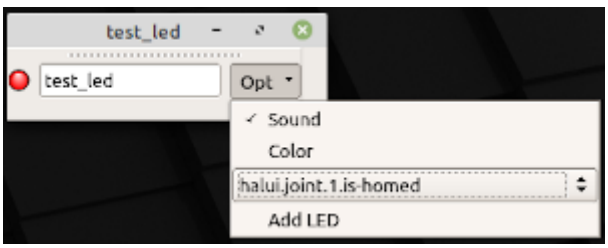


Figure 292. QtVCP `test_dial` Panel - Test LED VCP

`test_panel`

Sammlung nützlicher Widgets zum Testen von HAL-Komponenten, einschließlich der Sprach-Übermittlung des LED-Zustands.

```
loadusr qtvcp test_panel
```

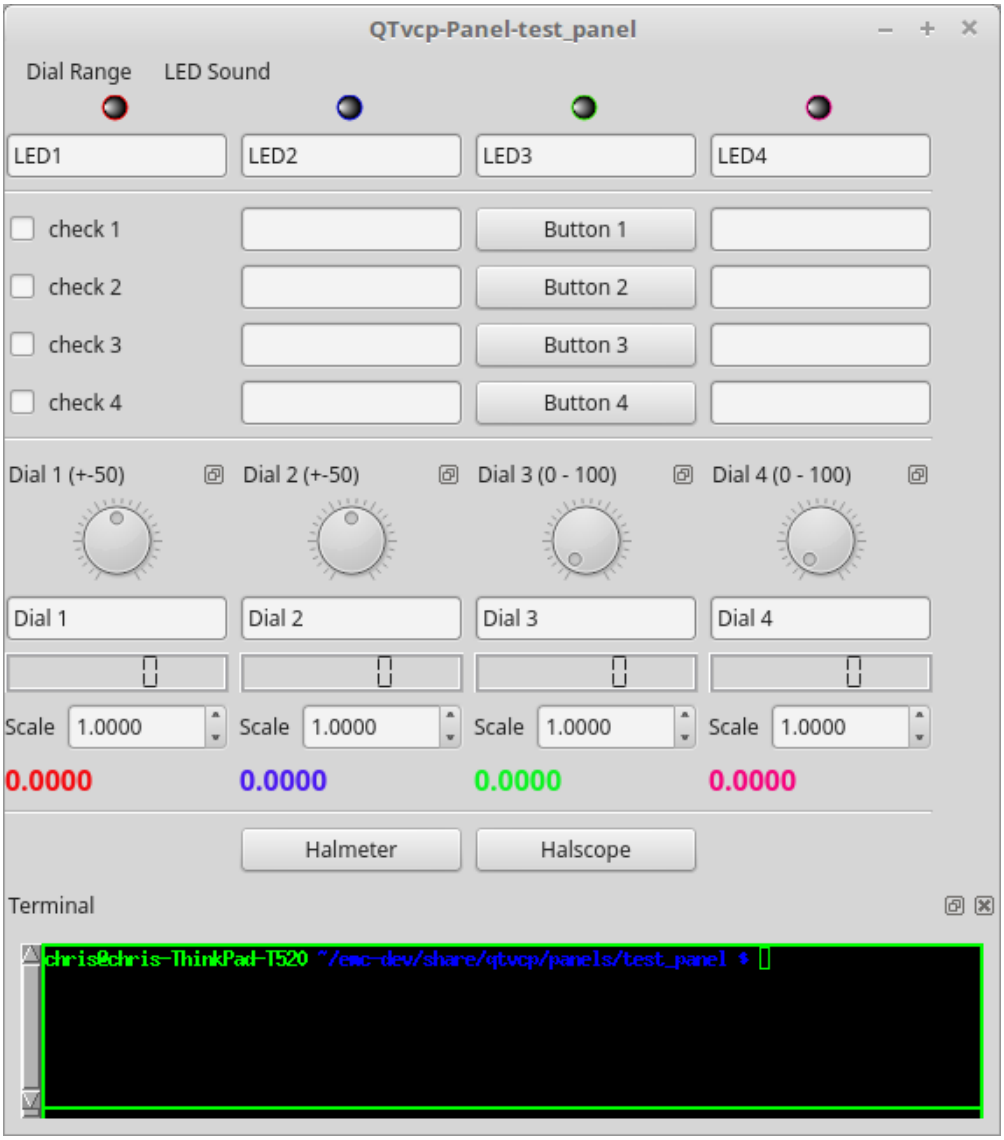


Figure 293. QtVCP `test_panel-` - HAL Komponenten-Prüffeld

`cam_align`

Ein **Kamera-Display-Widget für die Rotationsausrichtung**.

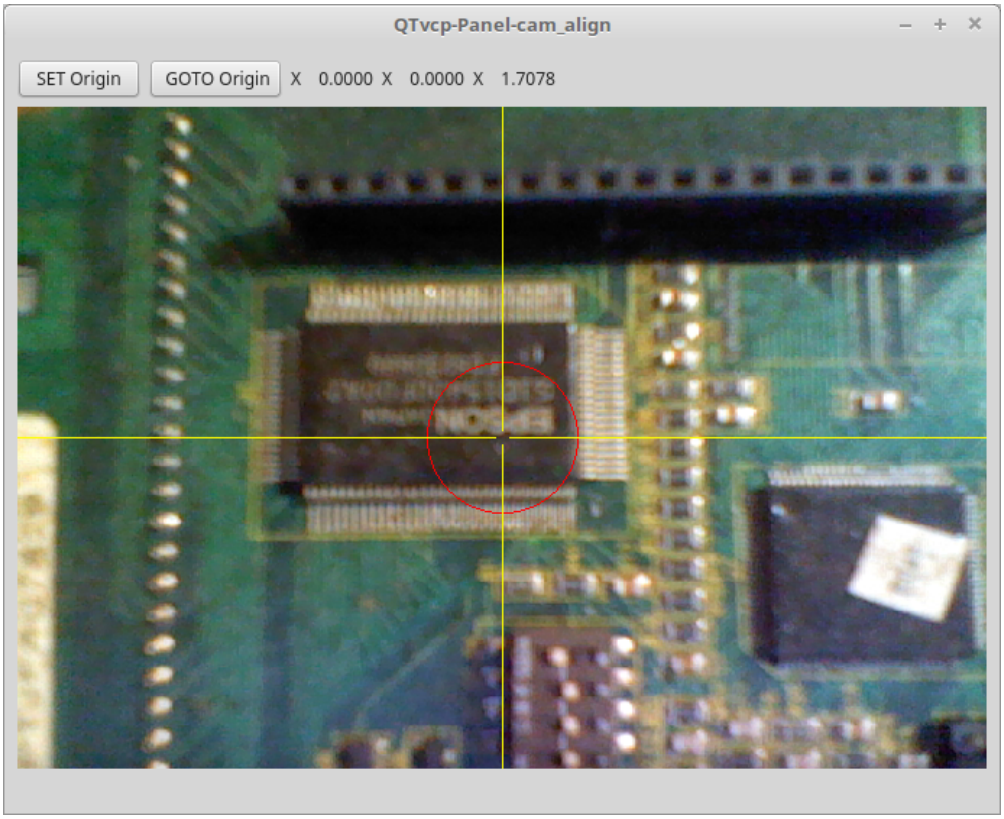


Figure 294. QtVCP `cam_align` Panel zur Kamera-basierten Ausrichtung

Anwendung

Fügen Sie der INI-Datei diese Zeilen hinzu:

```
[DISPLAY]
EMBED_TAB_NAME = cam_align
EMBED_TAB_COMMAND = halcmd loadusr -Wn qtvcp_embed qtvcp -d -c qtvcp_embed -x {XID}
cam_align
# The following line is needed if embedding in GMOCCAPY
EMBED_TAB_LOCATION = ntb_preview
```

NOTE Alle "<Optionen>" müssen vor dem `cam_align` Panel Namen stehen.

QtVCP Optionen

- `-c NAME` Name der HAL-Komponente. Standardmäßig wird der UI-Dateiname verwendet.
- `-d` Debugging aktiviert. Oder entfernen, um keine minimale Ausgabe zu erhalten
- `-v` Einschaltbares ausführliches Debugging. Damit lassen sich alle verfügbaren Auflösungen ermitteln.
- `-x {XID}` wird für die Einbettung in AXIS oder Gmoccapy verwendet
- `-o <option>` Optionen, die an cam_align übergeben werden. Mehrere -o Eingaben sind möglich.

Cam_align -Optionen

Dies sind die verfügbaren -o Optionen:

- `size=400,400` Größe des eingebetteten Fensters (Breite, Höhe)

- `imagesize=300,300` Größe des Bildes im Fenster (Breite, Höhe)
- `rotincr=5` Legt die Inkrementgröße der Fadenkreuzrotation fest (Grad)
- `xscale=100` Skaliert das Bild in X. Ein negativer Wert spiegelt das Bild in X (Prozent)
- `yscale=100` Skaliert das Bild in Y. Ein negativer Wert spiegelt das Bild in Y (Prozent)
- `camnumber=1` Legt fest, welche Systemkamera verwendet werden soll
- `api=V4L2` Legt die von OpenCV zu verwendende Backend-Bibliothek für die Kamera fest.
- `res=1280,720` Bestimmt die angeforderte Auflösung (Breite, Höhe)

Zu Beispiel können Breite und Höhe des Fensters, Drehung und Kameranummer in der INI mit der Option -o hinzugefügt werden.

```
EMBED_TAB_COMMAND = halcmd loadusr -Wn qtvcp_embed qtvcp -d -c qtvcp_embed -x {XID} -o
size=400,400 -o rotincr=.2 -o camnumber=0 cam_align
```

Maussteuerung:

- Linke Maus Einzelklick - Fadenkreuzrotation um einen Schritt erhöhen
- Einfacher Mausklick mit der rechten Maustaste - Drehung des Fadenkreuzes um eine Stufe verringern
- Einfacher Klick mit der mittleren Maustaste - zyklisch durch Rotationsschritte blättern
- linke Maustaste gedrückt halten und scrollen - Kamerazoom scrollen
- rechte Maustaste gedrückt halten und scrollen - Fadenkreuz-Drehwinkel scrollen
- nur Mausbewegung - Scroll bestimmt Kreis-Durchmesser
- Linker Maus-Doppelklick - Zoom zurücksetzen
- Doppelklick mit der rechten Maustaste - Rotation zurücksetzen
- Doppelklick mit der mittleren Maus - Kreisdurchmesser zurücksetzen

Um die oberen Schaltflächen zu verwenden, müssen Sie einen Befehl (oder eine Unterroutine) zuweisen. Dies könnte wie folgt aussehen:

```
[MDI_COMMAND_LIST]
MDI_COMMAND_CAM_ALIGN1=G10 L20 P1 X0 Y0,Set XY\nOrigin
MDI_COMMAND_CAM_ALIGN2=G0 X0 Y0,Go To\nOrigin
```

Dabei bezieht sich der erste Befehl auf die Schaltfläche "SET Ursprung" (engl. origin) und der zweite auf die Schaltfläche "GOTO Ursprung".
Beachten Sie, dass das Komma und der Text danach optional sind - sie überschreiben den Standardtext der Schaltfläche.
Diese Schaltflächen sind QtVCP-Aktionsschaltflächen und folgen diesen Regeln.

sim_panel

Kleines Bedienfeld, um **Schnellauf Handsteuerungen** für simulierte Konfigurationen **zu simulieren**.

Das Handsteuergerät (engl. kurz MPG), Auswahltasten und Steuertasten exportieren HAL-Pins, um sich mit linuxcnc zu verbinden.
Die Auswahl- und Kontrollgruppen-Boxen können durch die Option *-o hide=* verdeckt werden.
groupBoxControl und *groupBoxSelection* sind die Widget-Namen, die versteckt werden können.
Wenn Sie beide verbergen möchten, verwenden Sie eine Komma zwischen ihnen ohne Leerzeichen.
Die Option *-a* verhindert ein Verdecken des Panels durch andere Fenster.

```
loadusr qtvcp sim_panel
```

Hier laden wir das Panel ohne MPG-Auswahltasten und die immer-im-Vordergrund (engl. always-on-top)-Option.

```
loadusr qtvcp -a -o hide=groupBoxSelection sim_panel
```



Figure 295. QtVCP `sim_panel` - Simuliertes Bedienfeld für Bildschirmtests.

`tool_dialog`

Manueller Werkzeugwechsel-Dialog, der eine Werkzeugbeschreibung enthält.

```
loadusr -Wn tool_dialog qtvcp -o speak_on -o audio_on tool_dialog
```

Optionen:

- -o notify_on` - _verwendet Desktop-Notify-Dialoge anstelle der QtVCP-eigenen Dialoge.
- `-o audio_on` - *Ton beim Werkzeugwechsel abspielen*
- -`o talk_on` - *Sprach-Ankündigung des Werkzeugwechsels*

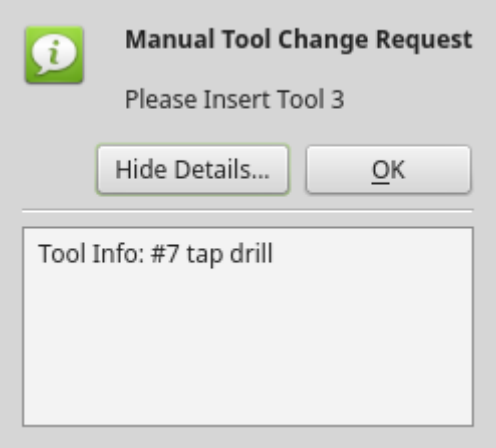


Figure 296. QtVCP `tool_dialog` - Dialog zum manuellen Werkzeugwechsel

12.7.2. `vismach` 3D Simulation Panels

Diese Tafeln sind vorgefertigte Simulationen gängiger Maschinentypen.

Diese können auch in andere Bildschirme wie AXIS oder GMOCCAPY eingebettet werden.

QtVCP `vismach_mill_xyz`

3D-OpenGL-Ansicht einer *3-Achsen-Fräsmaschine*.

```
loadusr qtvcp vismach_mill_xyz
```

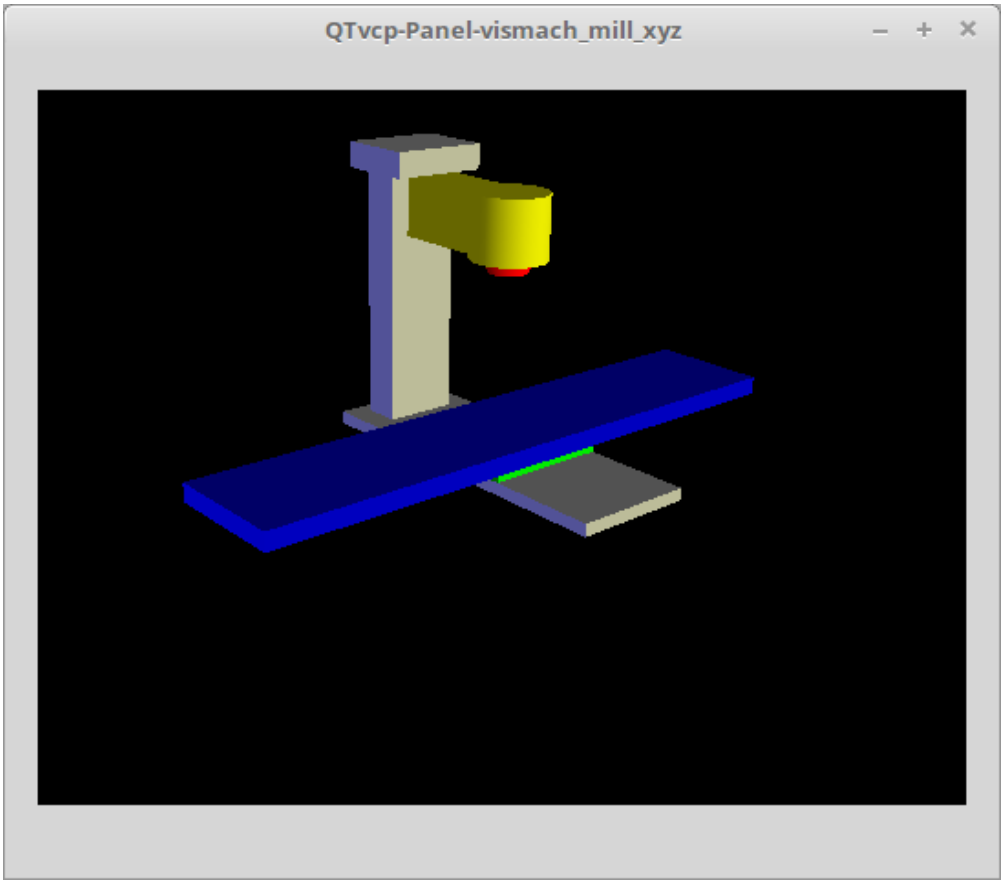


Figure 297. QtVCP `vismach_mill_xyz` - 3-Achsen-Fräse 3D-Ansichtspanel

QtVCP `vismach_router_atc`

3D-OpenGL-Ansicht einer *3-Achsen-Fräsmaschine im Gantry-Bett-Stil*.
Dieses spezielle Panel zeigt, wie man die Modellteile in der Handler-Datei definiert und verbindet, anstatt das vorgefertigte Modell aus der Vismach-Bibliothek von QtVCP zu importieren.

```
loadusr qtvcp vismach_router_atc
```

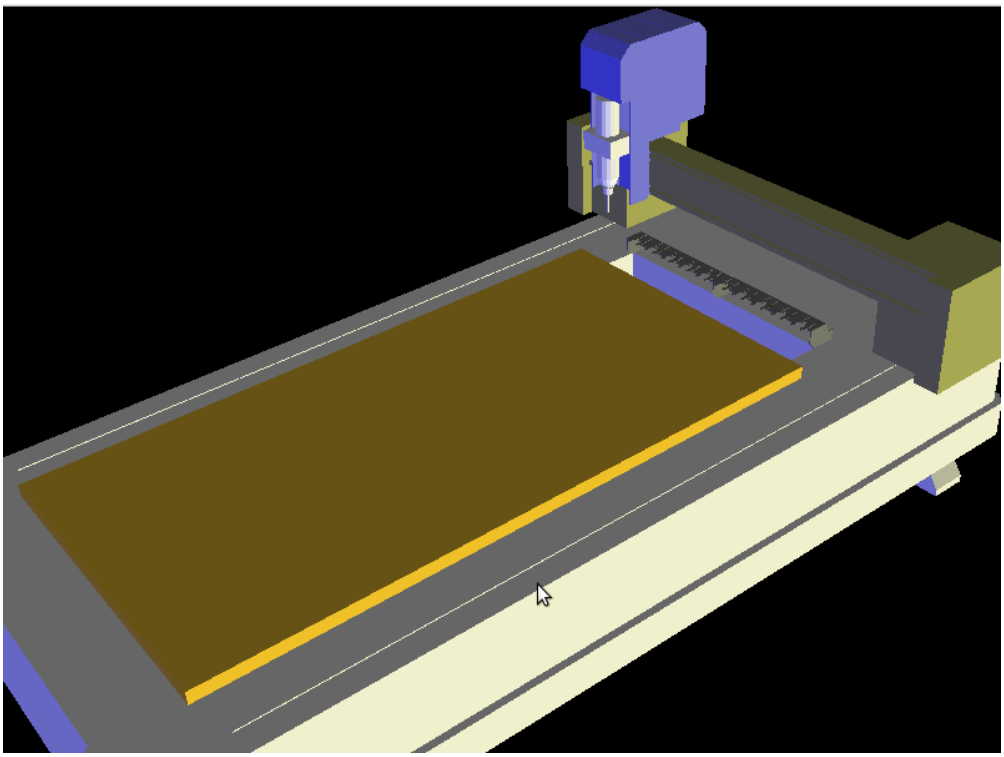


Figure 298. QtVCP `vismach_router_atc` - 3-Achsen Portalfräsen 3D-Ansichtspanel

QtVCP `vismach_scara`

3D-OpenGL-Ansicht einer *SCARA-basierten Fräsmaschine.*

```
loadusr qtvcp vismach_scara
```

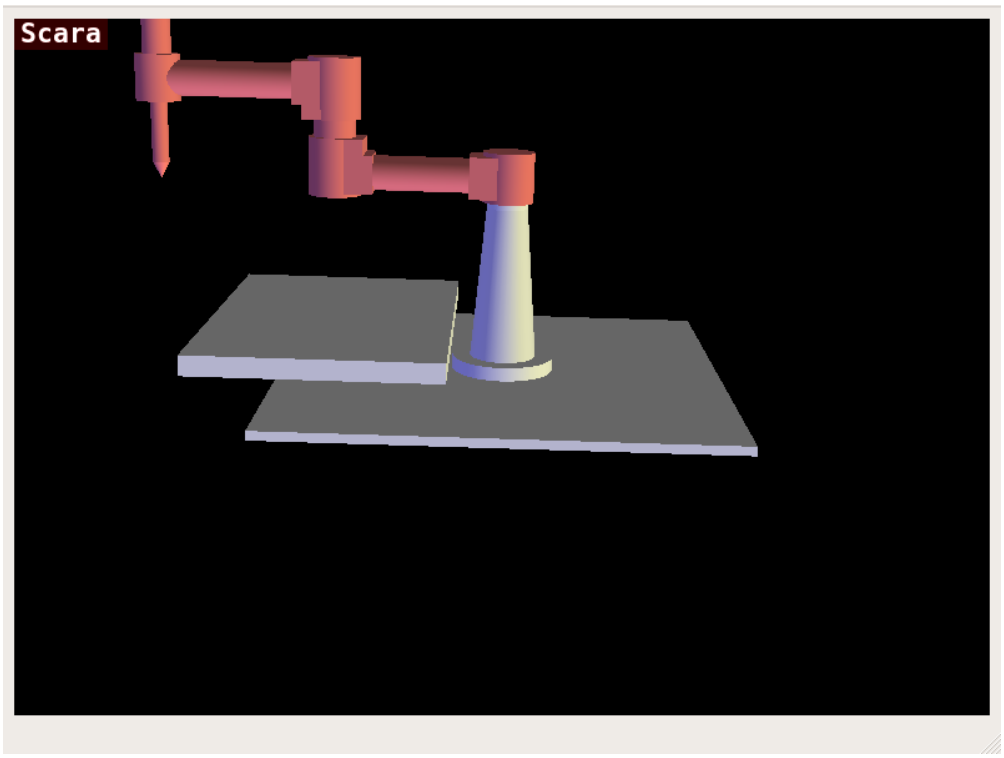


Figure 299. QtVCP `vismach_scara` - SCARA Fräse 3D-Ansichtsfenster

QtVCP `vismach_millturn`

3D OpenGL-Ansicht einer *3-Achsen-Fräsmaschine mit einer A-Achse/Spindel.*

```
loadusr qtvcp vismach_millturn
```

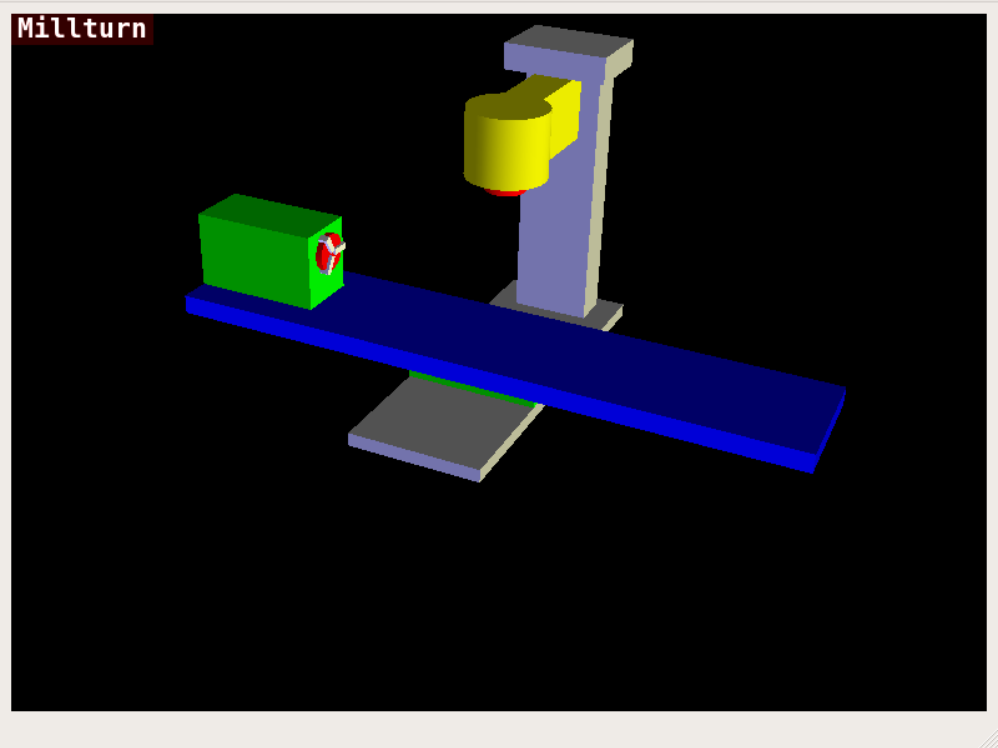


Figure 300. QtVCP `vismach_millturn` - 4 Achsen Drehzentrum 3D Ansichtspanel

QtVCP `vismach_mill_5axis_gantry`

3D-OpenGL-Ansicht einer *5-Achsen-Fräsmaschine in Gantry-Bauweise.*

```
loadusr qtvcp vismach_mill_5axis_gantry
```

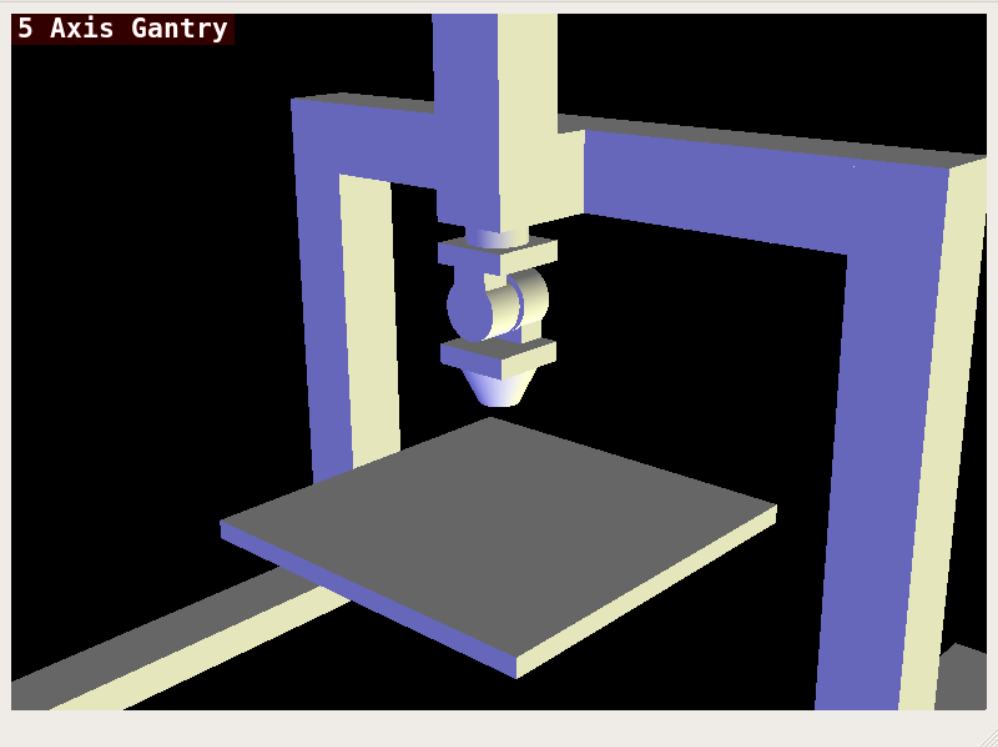


Figure 301. QtVCP `vismach_mill_5axis_gantry` - 5-Achsen Gantry Mill 3D-Ansichtspanel

QtVCP `vismach_fanuc_200f`

3D openGL-Ansicht eines *6-Gelenk-Roboterarms*.

```
loadusr qtvcp vismach_fanuc_200f
```

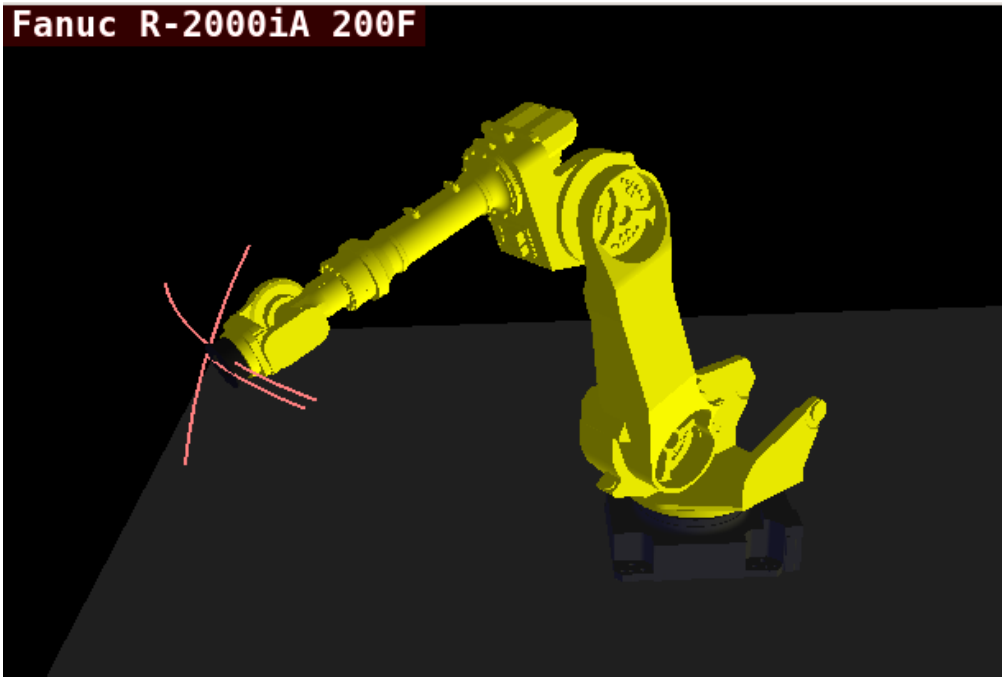


Figure 302. QtVCP `vismach_fanuc_200f` - 6-Gelenk-Roboterarm

12.7.3. Benutzerdefinierte virtuelle Schalttafeln (engl. panels)

Sie können natürlich **Ihr eigenes Panel erstellen und laden**.

Wenn Sie eine UI-Datei mit dem Namen `my_panel.ui` und eine HAL-Datei mit dem Namen `my_panel.hal` erstellt haben, würden Sie diese dann von einem Terminal aus laden mit:

```
halrun -I -f my_panel.hal
```

Beispiel einer HAL-Datei, die ein QtVCP-Panel lädt

```
# Echtzeitkomponenten laden
loadrt threads
loadrt classicladder_rt

# Nicht-Echtzeit-Programme laden
loadusr classicladder
loadusr -Wn my_panel qtvcp my_panel.ui  ①

# Komponenten zum Thread hinzufügen
addf classicladder.0.refresh thread1


# Pins verbinden
net bit-input1 test_panel.checkbox_1 classicladder.0.in-00
net bit-hide test_panel.checkbox_4 classicladder.0.hide_gui

net bit-output1    test_panel.led_1             classicladder.0.out-00

net s32-in1        test_panel.doublescale_1-s   classicladder.0.s32in-00

# start thread
start
```

① In diesem Fall laden wir `qtvcp` mit **`-Wn`**, das wartet, bis das Panel das Laden beendet hat, bevor es mit der Ausführung des nächsten HAL-Befehls fortfährt.
Damit soll *gewährleistet werden, dass die vom Panel erstellten HAL-Pins tatsächlich fertig sind*, falls sie im Rest der Datei verwendet werden.

12.7.4. Einbettung von QtVCP Virtual Control Panels in QtVCP-Bildschirme

QtVCP-Panels können in die meisten QtVCP-Bildschirme eingebettet werden und vermeiden Probleme wie z. B. Fokusübertragungen (engl. focus transferring), die bei nicht-nativen Einbettungen ein Problem darstellen können.

Befehle einbetten

Ein typischer Bildschirm wie QtDragon sucht in der INI-Datei unter der Überschrift [DISPLAY] nach Befehlen zum Einbetten eines Panels.

```
[DISPLAY]
EMBED_TAB_NAME=Embedding demo
```

```
EMBED_TAB_COMMAND=qtvcp simple_hal
EMBED_TAB_LOCATION=tabWidget_utilities
```

EMBED_TAB_NAME

ist normalerweise der Titel der Registerkarte.

EMBED_TAB_LOCATION

ist spezifisch für den Bildschirm und gibt das tabWidget oder stackedWidget an, in das eingebettet werden soll.

EMBED_TAB_COMMAND

ist der Befehl, mit dem das Laden des Panels aufgerufen wird. Bei nativen eingebetteten Panels ist das erste Wort immer "qtvcp", das letzte Wort ist der Name des zu ladenden Panels. Sie können auch Optionen an das Panel weitergeben mit den -o Schaltern auf Befehlszeile einfügen zwischen *qtvcp* und dem Panel-Namen. Das Panel folgt der Einstellung des Debugging-Modus des Hauptbildschirms.

Pfad zu mitgelieferten Panels

Es gibt Panels, die in LinuxCNC enthalten sind. Um eine Liste zu sehen, öffnen Sie ein Terminal, geben Sie "qtvcp" ein und drücken Sie die Eingabetaste.
Sie erhalten einen Hilfeausdruck und eine Liste der eingebauten Bildschirme und Bedienfelder.
Wählen Sie einen der Namen aus der Liste und fügen Sie ihn dem COMMAND-Eintrag nach *qtvcp* hinzu.
Der Suchpfad für die eingebauten Bedienfelder lautet *share/qtvcp/panels/PANELNAME*.
Run-In-Place und installierte Versionen von LinuxCNC haben diese an unterschiedlichen Stellen im System.

Pfad zu benutzerdefinierten Bedienfeldern

Auch benutzerdefinierte Panels können eingebettet werden - entweder ein modifiziertes eingebautes Panel oder ein neues vom Benutzer erstelltes.
Beim Laden von Panels sucht QtVCP im Pfad des Konfigurationsordners nach *qtvcp/panels/PANELNAME/PANELNAME.ui*.
PANNELNAME' ist ein beliebiger gültiger String ohne Leerzeichen. Wenn dort kein Pfad gefunden wird, dann wird im eingebauten Dateipfad gesucht.
QtVCP wird den gleichen Prozess für die optionale Handler-Datei durchführen: *qtvcp/panels/PANELNAME/PANELNAME_handler.py*

Tipps zur Handler-Programmierung

In einer Screen-Handler-Datei ist die Referenz für das Fenster "self.w".
In QtVCP-Panels bezieht sich diese Referenz auf das Fenster des Panels.
Um das Hauptfenster zu referenzieren, verwenden Sie *self.w.MAIN*. Wenn Ihr Panel unabhängig und eingebettet laufen soll, müssen Sie Fehler abfangen, die durch die Referenzierung von nicht verfügbaren Objekten entstehen. (Bemerkung: Objekte des Hauptbildschirms sind in einem unabhängigen Panel nicht verfügbar.)

Dies würde z.B. die Einstellungsdatei für den Panel verwenden, sofern eine solche vorhanden ist.

```
try:
    belt_en = self.w.PREFS_.getpref('Front_Belt_enabled', 1, int, 'SPINDLE_EXTRAS')
except:
    belt_en = 1
```

Dabei wird die Einstellungsdatei für den Hauptbildschirm verwendet, sofern eine solche vorhanden ist.

```
try:
    belt_en = self.w.MAIN.PREFS_.getpref('Front_Belt_enabled', 1, int, 'SPINDLE_EXTRAS')
except:
    belt_en = 1
```

Designer-Widget-Tipps

Bei Verwendung der Python-Befehlsoption in Action-Button-Widgets eines eingebetteten Panels:

`INSTANCE`

bezieht sich auf das Panel-Fenster, z. B. `INSTANCE.my_panel_handler_function_call(True)`

MAIN_INSTANCE

bezieht sich auf das Hauptbildschirmfenster (engl. main screen window). Z.B.: `MAIN_INSTANCE.my_main_screen_handler_function_call(True)`

Wenn das Panel nicht eingebettet ist, beziehen sich beide auf das Panel-Fenster.

Handler Patching – Unterklassenbildung eingebauter Bedienfelder

QtVCP kann eine untergeordnete Version der Standard-Handler-Datei laden. In dieser Datei können die ursprünglichen Funktionen verändert oder neue hinzugefügt werden.
Unterklassifizierung bedeutet lediglich, dass unsere Handler-Datei zuerst die originale Handler-Datei lädt und dann unseren neuen Code darüberlegt - im Wesentlichen eine Art Patch.
Dies ist nützlich, um Verhalten zu ändern oder zu erweitern, während weiterhin Updates der Standard-Handler aus den LinuxCNC-Repositories übernommen werden.

Möglicherweise müssen Sie weiterhin den Handler-Kopierdialog verwenden, um die Originaldatei des Handlers zu kopieren und zu entscheiden, wie Sie sie patchen.

Es sollte einen Ordner im Konfigurationsordner geben; für Panel: benannt als *<KONFIGURATIONSORDNER>/qtvcp/panels/<PANEL NAME>/*
Fügen Sie dort die Handler-Patch-Datei hinzu, benannt als <ORIGINAL PANEL NAME>_handler.py, demnach würde für QtDragon die Datei *cam_align_handler.py* genannt werden.

Hier ist ein Beispiel-Stylesheet zum Ändern der Farbe des Kreises in cam_align:

```
import sys
import os
import importlib
from PyQt5.QtCore import Qt
from qtvcp.core import Path
```

```
PATH = Path()

# erhalte Referenz zu original Handler Datei, um sie zu spezialisieren (eine Unterklasse
zu bilden, engl. to subclass)
sys.path.insert(0, PATH.PANELDIR)
panel = os.path.splitext(os.path.basename(os.path.basename(__file__)))[0]
base = panel.replace('_handler','')
module = "{}.{}".format(base,panel)
mod = importlib.import_module(module, PATH.PANELDIR)
sys.path.remove(PATH.PANELDIR)
HandlerClass = mod.HandlerClass

# return our subclassed handler object to Qtvcp
def get_handlers(halcomp, widgets, paths):
    return [UserHandlerClass(halcomp, widgets, paths)]

# spezialisiert (engl. subclassed) von HandlerClass die zuvor importiert wurde
class UserHandlerClass(HandlerClass):
    print('Custom subclassed panel handler loaded\n')

    def initialized__(self):
        # call original handler initialized function
        super().initialized__()

        # hinzufügen der eigenen Anpassung
        self.w.camview.circle_color = Qt.green
```

12.8. QtVCP Widgets

Qtscreen verwendet *QtVCP Widgets* für die LinuxCNC Integration.

Widget ist der allgemeine Name für die *UI-Objekte* wie Buttons und Beschriftungen in PyQt.

Es stehen Ihnen alle **Standard-Widgets** im *Qt Designer* Editor zur Verfügung.

Es gibt auch **spezielle Widgets** für LinuxCNC, um die Integration zu erleichtern. Diese sind in zwei Teile geteilt, überschrieben wie folgt auf der rechten Seite des Editors:

- Einer ist für **nur HAL-Widgets**.
- Das andere ist für **CNC-Steuerungs-Widgets**.

Es steht Ihnen frei, sie auf Ihrer Tafel beliebig zu mischen.

NOTE

Diese Beschreibung der Widget-Eigenschaften kann aufgrund der weiteren Entwicklung und des Mangels an Personen, die Dokumentationen schreiben, leicht veraltet sein (eine gute Möglichkeit, dem Projekt etwas zurückzugeben). Die endgültigen Beschreibungen finden Sie im Quellcode.

12.8.1. Nur HAL-Widgets

Diese Widgets haben normalerweise *HAL-Pins* und **reagieren nicht auf die Maschinensteuerung**.

`CheckBox` Widget

Mit diesem Widget kann der Benutzer **ein Kästchen ankreuzen, um einen HAL-Pin auf true oder false zu setzen**.

Er basiert auf dem *QCheckButton* von PyQt.

DetachTabWidget – Container-Widget mit vom Benutzer abtrennbaren Panels

Dieses Container-Widget funktioniert genau wie ein QTabWidget – es zeigt mehrere Panels nacheinander an, die über Tabs ausgewählt werden können.
Wenn man auf den Tab doppelklickt oder ihn zieht, löst er sich vom Hauptfenster.
Wird ein Tab abgetrennt, werden die Inhalte in einem QDialog platziert.
Der Tab kann wieder angehängt werden, indem man den Dialog schließt oder auf den Rahmen des Dialogfensters doppelklickt.

Es basiert auf PyQts QTabWidget.

`DoubleScale` - Eingabe-Widget für die Drehschaltfläche (engl. spin button)

This widget is a **spin button entry** widget used for *setting a sint and real HAL pin*.

Es hat einen internen *Skalierungsfaktor*, der standardmäßig auf 1 gesetzt ist und programmatisch oder über ein QtSignal eingestellt werden kann.

The `setInput` slot can be connected to an integer, or a real signal.

`[HALLabelName].setInput(some_value)`

Dies ist ein Funktionsaufruf zur Änderung des internen Skalierungsfaktors.

Die HAL-Pins werden auf den Wert der *internen Skala mal dem vom Widget angezeigten Wert* gesetzt.

`FocusOverlay` - Focus Overlay Widget

Dieses Widget legt ein **farbiges Overlay über den Bildschirm**, normalerweise während ein Dialog angezeigt wird.

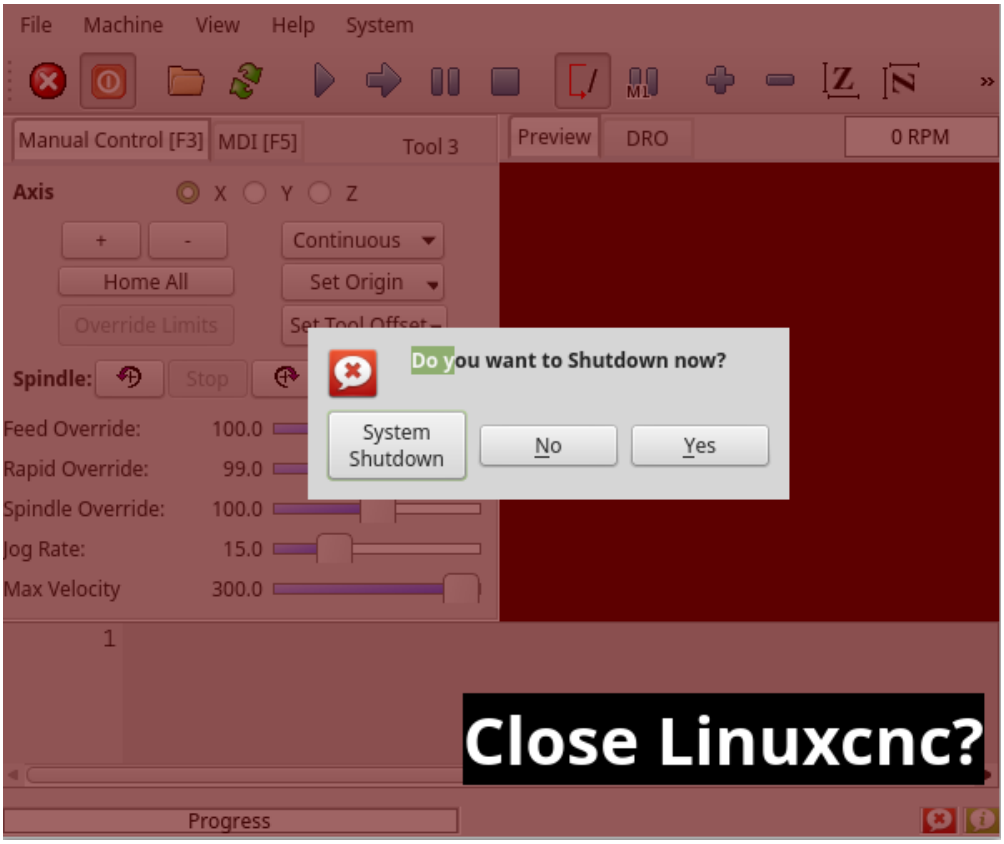


Figure 303. Beispiel für ein Fokus-Overlay zur Bestätigung der Abschlussaufforderung

Wird verwendet, um ein "konzentriertes" Gefühl zu erzeugen und die Aufmerksamkeit auf wichtige Informationen zu lenken.

Es kann auch ein durchsichtiges Bild anzeigen.
Es kann auch Nachrichtentext und Schaltflächen anzeigen.

Dieses Widget *kann mit 'STATUS'-Meldungen gesteuert werden.*

Gauge - Rundes Messuhr-Widget

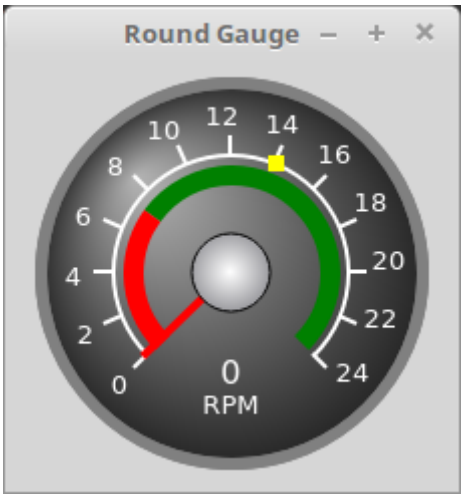


Figure 304. QtVCP Gauge: Rundes Messuhr-Widget

Round Gauge kann in einem LinuxCNC GUI verwendet werden, um **einen Eingabeparameter** auf dem Zifferblatt anzuzeigen.

Anpassbare Parameter

Es gibt mehrere Eigenschaften, die vom Benutzer eingestellt werden können, um das *Erscheinungsbild der Anzeige* anzupassen.

Die folgenden Parameter können entweder programmatisch oder über den Eigenschaftseditor von Qt Designer eingestellt werden.

`halpin_option`

Wenn Sie diese Option auf `True` setzen, werden *2 HAL-Pins* erstellt:

- Einer ist für die Eingabe des `value` (engl. für Wert) bestimmt
- Die andere dient der Einstellung des "Sollwerts" (engl. `setpoint`).

Wenn diese Option nicht gesetzt ist, müssen `value` und `setpoint` programmatisch, d.h. in der Handler-Datei, verbunden werden.

`max_reading`

Dieser Wert bestimmt die *höchste angezeigte Zahl* auf der Anzeige.

`max_value`

Dies ist der *maximal zu erwartende Wert des Werteingangssignals*.
Mit anderen Worten, es ist der Skalenendwert.

`num_ticks`

Dies ist die *Anzahl der Ticks/Anzeigewerte* auf der Anzeigefläche.
Sie sollte auf eine Zahl eingestellt werden, die sicherstellt, dass die Textanzeigen auf der Anzeigefläche lesbar sind.
Der minimal zulässige Wert ist 2.

`zone1_color`

Zone1 erstreckt sich *vom maximalen Messwert bis zum Schwellenwert*.
Sie kann auf eine beliebige RGB-Farbe eingestellt werden.

`zone2_color`

Zone2 erstreckt sich *vom Schwellenwert bis zum Mindestwert*, der 0 ist.
Sie kann auf eine beliebige RGB-Farbe eingestellt werden.

`bezel_color`

Dies ist die Farbe des *Außenrings des Messgeräts*.

`bezel_width`

Dies ist die Stärke (engl. width) des *Außenrings des Messgeräts*.

`threshold`

Der Schwellenwert ist der *Übergangspunkt zwischen den Zonen*.
Er sollte auf einen Wert zwischen 0 und dem Maximalwert gesetzt werden.
Der höchstzulässige Wert wird auf den "Maximalwert" des Messgeräts gesetzt, der Mindestwert ist 0.

gauge_label

Dies ist der *Text unter der Wertanzeige*, nahe dem unteren Rand des Messgeräts.
Die Funktion des Messgeräts ist dann leicht erkennbar.

base_color

Farbe der Mess-Anzeige (engl. gauge).

base_gradient_color

Die Hervorhebungsfarbe der Anzeige.

center_color

Dies Farbe des *Zentrums der Messanzeige*.

center_gradient_color

Dies Hervorhebungsfarbe des *Zentrums der Messanzeige*.

Nicht anpassbare Parameter

Es gibt 2 Eingänge, die nicht anpassbar sind. Sie können über HAL-Pins, programmatisch oder über Signale von anderen Widgets gesetzt werden:

value

Dies ist der *eigentliche Eingangswert*, der mit der Nadel des Messgeräts und in der digitalen Anzeige angezeigt wird.
Er muss auf einen Wert zwischen 0 und dem Maximalwert eingestellt werden.

setpoint

Dies ist ein Wert, der die Position einer kleinen *Markierung auf der Messgeräteoberfläche* bestimmt.
Er muss auf einen Wert zwischen 0 und dem Maximalwert gesetzt werden.

GeneralHALInput - Allgemeine Signale/Steckplätze (engl. slot) Eingang Verbindung Widget

Dieses Widget wird verwendet, um **ein beliebiges Qt-Widget über Signale/Slots** mit HAL zu verbinden.

Sie wird _für Widgets verwendet, die auf HAL-Pin-Änderungen **reagieren** sollen.

GeneralHALOutput - Allgemeines Signal/Slot-Ausgangsverbindungs-Widget

Dieses Widget wird verwendet, um **ein beliebiges Qt-Widget über Signale/Slots** mit HAL zu verbinden.

Es wird _für Widgets verwendet, die HAL-Pins **steuern** sollen.

GridLayout - Grid Layout Widget

Dieses Widget *steuert*, ob die Widgets darin aktiviert oder deaktiviert sind.

Deaktivierte Widgets haben normalerweise eine andere Farbe und reagieren nicht auf Aktionen.

Es basiert auf dem `QGridLayout` von PyQt.

HalBar - HAL Bar Level Anzeige

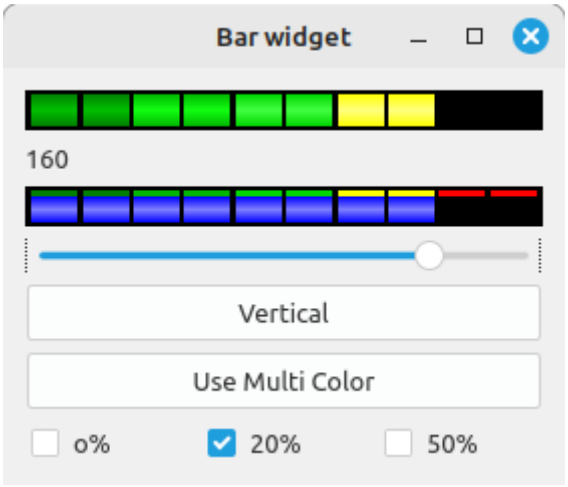


Figure 305. QtVCP `HalBar`: Panel zeigt die HAL Bar Level Anzeige

This widget is used to indicate level or value, usually of a HAL sint/real pin.
You can also disable the HAL pin and use Qt signals or Python commands to change the level.

Balken (engl. bar) Eigenschaften

HalBar ist eine Unterklasse des Bar (engl. für Balken) Widget, so dass es diese Eigenschaften erbt:

- *stepColorList*: eine Liste von Farbketten, die Anzahl der Farben definiert die Anzahl der Balken.
- *backgroundColor*: eine QColor Definition der Hintergrundfarbe.
- *indicatorColor*: eine QColor Definition der optoinalen alleinigen Farbwert der bar.
- *useMultiColorIndicator*: Boole'scher-Schalter zur Auswahl der Option einer einfarbigen oder mehrfarbigen Wertanzeige.
- *split*: der ganzzahlige prozentuale Anteil zwischen der Maximalwert-Anzeige und der aktuellen Wert-Anzeige (0 bis 50%).
- 'setVerticals: boolean Schalter zur Auswahl einer vertikalen oder horizontalen Anzeige.
- 'setInverteds: boolean Schalter zur Festlegung einer invertierten Richtung.
- *setMaximum*: eine ganze Zahl, die den maximalen Indikationsgrad (engl. level of indication) definiert.
- *setMinimum*: eine ganze Zahl, die den niedrigsten Indikationsgrad festlegt.

HalBar Eigenschaften

- *pinType*: Zur Auswahl des **HAL-Pin Typen**:
  - `NONE` kein HAL-Pin wird hinzugefügt
  - `SINT` A sint integer pin will be added
  - `REAL` A real pin will be added

- *pinName*: Um den **HAL-Pinnamen** zu ändern, sonst wird der Widget-Basisname verwendet.

HalBar Stylesheets

Die oben genannten Eigenschaften für Balken (engl. bar) können in *Stylesheets* festgelegt werden. pinType und pinName Eigenschaften können nicht in Stylesheets geändert werden.

NOTE	In Stylesheets ist stepColorList eine einzelne Zeichenfolge von Farbnamen, die durch Kommas getrennt sind.

```
HalBar{
    qproperty-backgroundColor: #000;
    qproperty-stepColorList:
'green,green,#00b600,#00b600,#00d600,#00d600,yellow,yellow,red,red';
}
```

HALPad - HAL Buttons Joypad

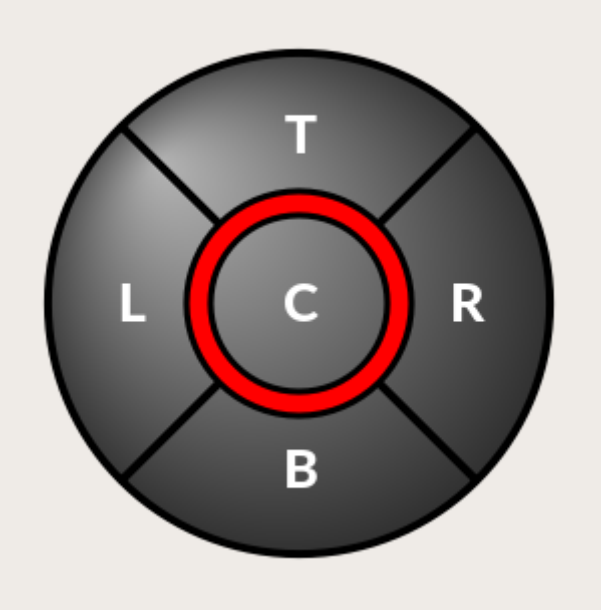


Figure 306. QtVCP HALPad: HAL-Buttons Joypad

Dieses Widget sieht aus und funktioniert wie ein **5-Tasten-D-Pad**, mit einem LED-Ring.

Each button has an selectable type (BOOL, SINT or REAL) output HAL pin.

The LED center ring has selectable colors for off and on and is controlled by a bool HAL pin.

HALPad ENUMS

Es werden *numerierte Konstanten* verwendet:

- Um **Indikatorpositionen** zu referenzieren:
  - NONE
  - LEFT

- RIGHT
- CENTER
- TOP
- BOTTOM
- LEFTRIGHT
- TOPBOTTOM

- Für **HAL-Pins Typ**:
  - NONE
  - BOOL
  - SINT
  - REAL

Sie verwenden den Namen des Widgets im Qt Designer plus die Referenzkonstante:

```
self.w.halpadname.set_highlight(self.w.halpadname.LEFTRIGHT)
```

HALPad *Eigenschaften*

pin_name

Optionaler Name, der für den *HAL Pin basename* verwendet wird. Bleibt er leer, wird der Name des Qt Designer Widgets verwendet.

pin_type

Wählen Sie den *HAL-Ausgangspin-Typ*. Diese Eigenschaft wird nur beim Starten verwendet. Die Auswahl kann im Qt Designer eingestellt werden:

- NONE
- BOOL
- SINT
- REAL

left_image_path

right_image_path

center_image_path

top_image_path

bottom_image_path

Datei- oder Ressourcenpfad zu einem Bild, das an der beschriebenen Stelle der Schaltfläche angezeigt werden soll.
Wenn die Schaltfläche "Zurücksetzen" (engl. reset) in der Qt Designer-Editor-Eigenschaft gedrückt wird, wird das Bild nicht angezeigt (optionaler Text ist möglich).

left_text

right_text

center_text

top_text

bottom_text

Eine Textzeichenfolge, die an der beschriebenen Schaltflächenposition angezeigt werden soll. Wenn das Feld leer gelassen wird, kann ein Bild für die Anzeige bestimmt werden.

true_color

false_color

Farbauswahl für den mittleren LED-Ring, der angezeigt werden soll, wenn der `BASENAME.light.center` HAL-Pin `True` oder `False` ist.

text_color

Auswahl der Farbe für den Text des Buttons.

text_font

Auswahl der Schriftart für den Text des Buttons.

`HALPad` *Styles*

Die oben genannten Eigenschaften können in *Stylesheets* festgelegt werden.

```
HALPad{
    qproperty-on_color: #000;
    qproperty-off_color: #444;
}
```

`HALLabel` - HAL Label Widget

Dieses Widget **zeigt die an dieses gesendeten Werte an**.

Werte können gesendet werden von:

- *HAL pins*
  The input pin can be selected as BOOL, SINT, REAL or no pin selected
- *Programmatisch*
- *Ein `QtSignal`*

Es gibt eine "textTemplate"-Eigenschaft, um den Rich-Text einzustellen und/oder den Text zu formatieren.
Eine grundlegende Formatierung könnte sein:

- `%r` für Boolesche Werte
- `%d` für ganze zahlen
- `%0.4f` für Floats.

Ein Beispiel für einen Rich-Text könnte sein:

```
self.w.my_hal_label.setProperty(textTemplate,"""
<html>
<head/>
<body>
  <p><span style="font-size:12pt;font-weight:600;color:#f40c11;">%0.4f</span></p>
</body>
</html>
"""
)
```

The `setDisplay` slot can be connected to an integer, a real or a bool signal.

Wenn die Eigenschaft `pin_name` nicht gesetzt ist, wird der Name des Widgets verwendet.

Es gibt Funktionsaufrufe zur Anzeige von Werten:

`[HALLabelName].setDisplay(some_value)`

Kann zur Einstellung der Anzeige (engl. display) verwendet werden, wenn kein HAL-Pin ausgewählt ist.

`[HALLabelName].setProperty(textTemplate,"%d")`

Legt die Vorlage für die Anzeige fest.

Es basiert auf PyQts *QLabel*.

`LCDNumber` - Widget zum Auslesen der LCD-Stilnummer

This widget *displays HAL real/sint/bool values in a LCD looking way*.

Es kann Zahlen im Dezimal-, Hexadezimal-, Binär- und Oktalformat anzeigen, indem es die Eigenschaft **`Modus`** setzt.

Bei der Verwendung von Floats können Sie eine Formatierung über eine Zeichenfolge festlegen.

Sie müssen die Eigenschaft **`digitCount`** auf eine geeignete Einstellung setzen, um die größte Zahl anzuzeigen.

Eigenschaften

`pin_name`

Optionsstring, der als HAL-Pin-Name verwendet werden soll.
Bei einem leeren String wird der Name des Widgets verwendet.

`bit_pin_type`

Selects the input pin as type `BOOL`.

`s32_pin_type`

Selects the input pin as type `SINT`.

float_pin_type

Select the input pin as type REAL.

floatTemplate

A string that will be used as a Python3 format template to tailor the LCD display.
Only used when a REAL pin is selected, e.g., {:.2f} will display a float rounded to 2 numbers after the decimal.
A blank setting will allow the decimal to move as required.

Es basiert auf PyQts *QLCDNumber*.

LED - Anzeige-Widget

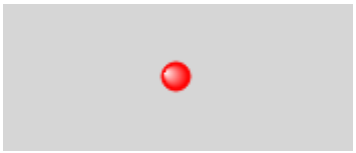


Figure 307. QtVCP LED: LED-Anzeige-Widget

Eine **LED-ähnliche Anzeige**, die optional der Logik eines HAL-Pins folgt.

halpin_option

Wählt aus, ob die LED einem Eingangs-HAL-Pin oder einem Programmzustand folgt.

diameter

Durchmesser der LED (voreingestellt auf 15).

color

Farbe der LED im eingeschalteten Zustand (Voreingestellt auf "green" (engl. für grün)).

off_color

Farbe der LED im ausgeschalteten Zustand (Voreingestellt auf black (engl. für schwarz)).

gradient

schaltet die Verlaufs-Hervorhebung ein oder aus (voreingestellt: ein).

on_gradient_color

Farbe der LED im eingeschalteten Zustand (Voreingestellt auf white (engl. für weiß)).

off_gradient_color

Farbe der LED im ausgeschalteten Zustand (Voreingestellt auf white (engl. für weiß)).

alignment

Qt-Hinweis zur Ausrichtung.

state

Aktueller Zustand der LED

`flashing`

Schaltet die Blinkoption ein und aus.

`flashRate`

Legt die Blitzrate fest.

Die `LED`-Eigenschaften können in einem *stylesheet* mit folgendem Code definiert werden, welcher der `.qss`-Datei hinzugefügt wird, wobei `name_of_led` der im Qt Designer-Editor definierte Widget-Name ist:

```
LED #name_0f_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
}
```

`PushButton` - HAL Pin Toggle Widget

Mit diesem Widget kann der Benutzer einen **HAL-Pin per Tastendruck auf "true" oder "false" setzen**.

Als Option kann es eine *Umschalttaste* (engl. toggle button) sein.

For a *LED Indicator Option*, see IndicatedPushButton below for more info.

Es gibt auch andere Optionen.

Es basiert auf PyQts *QPushButton*.

`RadioButton` Widget

Mit diesem Widget kann ein Benutzer HAL-Pins auf true oder false setzen. Nur ein "RadioButton"-Widget einer Gruppe kann gleichzeitig "true" sein.

Er basiert auf dem *QRadioButton* von PyQt.

`Slider` - HAL-Pin-Wert-Anpassungs-Widget

Ermöglicht das **Anpassen eines HAL-Pins mit einem verschiebbaren Zeiger**.

`TabWidget` - Tab Widget

Dieses Widget ermöglicht es, die Tabulatorhöhe mit Stylesheets anzupassen.

Die `TabWidget`-Eigenschaften können in einem *Stylesheet* definiert werden, indem der folgende Code zur `.qss`-Datei hinzugefügt wird.
`name_of_tab` bezeichnet den im Qt Designer-Editor definierten Widget-Namen.
Wenn Sie den Text *#name_of_tab* weglassen, wird die Tab-Höhe aller `TabWidget`s festgelegt.
Dieses Beispiel zeigt, wie die Tab-Höhe eines bestimmten Widgets eingestellt wird:

```
TabWidget #name_des_tab{
  qproperty-tabsize: 1.5;
}
```

Es basiert auf PyQts QTabWidget.

`WidgetSwitcher` - Multi-Widget-Layout-View-Switcher-Widget

Dies wird verwendet, um die Ansicht eines Multi-Widget-Layouts umzuschalten, um nur ein Widget anzuzeigen, d.h. um **zwischen einer großen Ansicht eines Widgets und einer kleineren Multi-Widget-Ansicht zu wechseln**.

Es unterscheidet sich von einem gestapelten Widget, da es ein Widget von einer beliebigen Stelle des Bildschirms ziehen und es auf seiner Seite mit einem anderen Layout als ursprünglich platzieren kann.

Das *original Widget muss sich in einem layout befinden*, damit der Switcher es wieder ablegen kann.

In Qt Designer werden Sie:

- Das `WidgetSwitcher` widget auf dem Bildschirm hinzufügen.
- Auf dem `WidgetSwitcher`rechts-klicken, um eine Seite hinzuzufügen.
- Es bevölkern mit den Widgets/Layouts, die Sie in einem Standardformular sehen möchten.
- So viele Seiten hinzufügen, wie es Ansichten zu wechseln gibt.
- Fügen Sie auf jeder Seite ein Layout-Widget hinzu.
  Nach dem Hinzufügen des Layouts müssen Sie erneut mit der rechten Maustaste auf den Widget-Wechsler klicken und die Layout-Option festlegen.
- Klicken Sie auf das Widget `WidgetSwitcher` und scrollen Sie dann zum unteren Rand des Eigenschaftseditors.
- Suchen Sie die dynamische Eigenschaft `widget_list` und doppelklicken Sie rechts daneben.
- Es erscheint ein Dialogfeld, in dem Sie die Namen der Widgets angeben können, die auf die Seiten verschoben werden sollen, die Sie dem `WidgetSwitcher` hinzugefügt haben.

Es gibt *Funktionsaufrufe*, um bestimmte Widgets anzuzeigen.
Durch den Aufruf einer dieser Funktionen steuern Sie, welches Widget gerade angezeigt wird:

`[_WidgetSwitcherName_].show_id_widget(_number_)`

`[_WidgetSwitcherName_].show_named_widget(_widget_name_)`

`[_WidgetSwitcherName_].show_default()`

Dies zeigt das "Seite 0"-Layout und stellt alle anderen Widgets wieder so ein, wie sie ursprünglich in Qt Designer erstellt wurden.

`[_WidgetSwitcherName_].show_next()`

Nächstes Widget anzeigen.

Es basiert auf dem *QStack*-Widget.

`XEmbed` - Widget zum Einbetten von Programmen

Ermöglicht die **Einbettung eines Programms in das Widget**.

Es funktionieren nur Programme, die das `xembed`-Protokoll verwenden, wie z.B.:

- GladeVCP Virtuelle Control Panels
- Integrierte virtuelle Tastatur
- QtVCP Virtuelle Kontrollpanels
- mplayer-Videoplayer

12.8.2. Widgets für Maschinensteuerungen

Diese Widgets **interagieren mit dem Zustand der Maschinensteuerung**.

`ActionButton` - Aktionssteuerungs-Widget der Maschinensteuerung

Diese Tasten werden für **Steuerungsaktionen an der Maschinensteuerung** verwendet.

Sie sind auf `IndicatedPushButton` aufgebaut und können daher mit LEDs überlagert werden.

NOTE	Wenn Sie mit der linken Maustaste auf dieses Widget doppelklicken, können Sie einen Dialog zum Einstellen einer dieser Aktionen aufrufen. Die Dialoge helfen dabei, die richtigen Daten für die ausgewählte Aktion festzulegen. Sie können diese Eigenschaften auch direkt im Eigenschaftseditor ändern.

Aktionen

Sie können eine der folgenden Optionen auswählen:

`Estop`

`Machine On`

`Auto`

`mdi`

`manual`

`run`

`run_from_line status`

Ermittelt die Zeilennummer aus der `STATUS`-Meldung `gcode-line-selected`.

`run_from_line slot`

Ermittelt die Zeilennummer aus dem Qt Designer int/str Slot `setRunFromLine`.

`abort`

`pause`

`load dialog`

Erfordert das Vorhandensein eines Dialog-Widgets.

Camview dialog

Erfordert das Vorhandensein des Dialog-Widgets "Camview".

origin offset dialog

Erfordert das Vorhandensein eines Dialogfensters für den Ursprungsversatz.

macro dialog

Erfordert das Vorhandensein eines Makro-Dialog-Widgets.

Launch Halmeter

Launch Status

Launch Halshow

Home

Setzen Sie die Gelenknummer auf -1 für `all-home` (engl. alle referenzieren).

Unhome

Setzen Sie die Gelenknummer auf -1 für `all-unhome`(engl. für alle Referezierungen aufheben).

Home Selected

Setzt das durch „STATUS“ ausgewählte Gelenk/Achse in die Ausgangsstellung.

Unhome Selected

Hebt die Referenzierung der `STATUS` ausgewählte Verbindung/Achse auf.

zero axis

zero G5X

Nullt die aktuellen Offsets des Benutzerkoordinatensystems.

zero G92

Nullt die optionalen `G92`-Offsets.

zero Z rotational

Setzt den Rotationsoffset auf Null.

jog joint positive

Legt die Gelenknummer fest.

jog joint negative

Legt die Gelenknummer fest.

jog selected positive

Ausgewählt mit einem anderen Widget oder `STATUS`.

jog selected negative

Ausgewählt mit einem anderen Widget oder `STATUS`.

`jog increment`

Metrische/imperiale/angulare Zahlen einstellen.

`jog rate`

Festlegen Sie die float/alt-Gleitkommanummer.

`feed override`

Festlegen Sie die float/alt-Gleitkommanummer.

`rapid override` (engl. für Eilgang-Anpassung)

Festlegen Sie die float/alt-Gleitkommanummer.

`spindle override` (engl. für Spindel-Anpassung)

Festlegen Sie die float/alt-Gleitkommanummer.

`spindle fwd` (engl. für Spindel vorwärts)

`spindle backward` (engl. für Spindel rückwärts)

`spindle stop`

`spindle up` (engl. für Spindel hoch)

`spindle down` (engl. für Spindel runter)

`view change` (engl. für Änderungen anschauen)

Setzen von *view_type_string*.

`limits override` (engl. für Grenzen neufestsetzen)

`flood` (engl. für das Flut-Kühlmittel)

`mist` (engl. für (Kühl-)Nebel)

`block_delete` (engl. für Block löschen)

`optional stop`

`mdi command` (engl. für MDI Befehl)

Setze `command_string`, d.h. ruft einen fest kodierten MDI-Befehl auf

INI-MDI-Nummer

Setzt `ini_mdi_number`, d.h. ruft einen INI-basierten MDI-Befehl auf

`dro absolute`

`dro relative`

`dro dtg`

Exit-Bildschirm

Beendet LinuxCNC

`Override limits`

Vorübergehende Überschreitung harter Grenzen

Dialoge starten

Öffnet Dialogfelder, wenn sie in der UI-Datei enthalten sind.

`set DRO to relative` (engl. für *DRO auf relativ setzen*)

`set DRO to absolute`

`set DRO to distance-to-go` (engl. für *DRO auf Restweg setzen*)

Attribute

Diese setzen *Attribute* der ausgewählten Aktion (Verfügbarkeit hängt vom Widget ab):

`toggle float option` (engl. für Wechsel-Buttons)

Ermöglicht das Umschalten zwischen zwei Raten.

`joint number` (engl. für Gelenknummer)

Wählt das Gelenk/die Achse aus, das/die von der Schaltfläche gesteuert wird.

`incr imperial number`

Legt das imperiale Jog-Inkrement fest (negativ setzen, um zu ignorieren).

`incr mm number`

Legt die metrische Schrittweite fest (zum Ignorieren negativ setzen).

`incr angular number`

Legt die Winkelschrittweite fest (zum Ignorieren negativ einstellen).

`float number`

Wird für `jograte` und overrides verwendet.

`float alternate number`

Für `jograte` und overrides, die zwischen zwei Fließkommazahlen wechseln können.

`view type string`

Kann sein:

- `p`,
- `x`, `y`, `y2`, `z`, `z2`,
- `zoom-in`, `zoom-out`,
- `pan-up`, `pan-down`, `pan-left`, `pan-right`,
- `rotate-up`, `rotate-down`, `rotate-cw`, `rotate-ccw`
- `clear`.

`command string`

MDI-Befehlszeichenfolge, die aufgerufen wird, wenn die MDI-Befehlsaktion ausgewählt wird.

`ini_mdi_number`

(ehemaliger (engl. legacy) Weg einer Umsetzung)
Ein Verweis auf den Abschnitt `[MDI_COMMAND_LIST]` der _INI-Datei.
Setzen Sie einen Integer, der eine Zeile unter der INI-Zeile `[MDI_COMMAND]` auswählt, beginnend bei 0.

Fügen Sie dann in der INI-Datei unter der Überschrift `[MDI_COMMAND_LIST]` entsprechende Zeilen hinzu.
Befehle werden durch `;` getrennt.
Die button-Beschriftung (engl. label) wird durch beliebigen Text nach dem Komma gesetzt, das Symbol `\n` fügt einen Zeilenumbruch hinzu.

`ini_mdi_key`

(bevorzugte Weise)
Ein Verweis auf den Abschnitt `[MDI_COMMAND_LIST]` in der *INI-Datei*.
Diese Zeichenkette wird zu *MDI_COMMAND_* hinzugefügt, um einen Eintrag zu bilden, nach dem in der INI-Datei unter der Überschrift `[MDI_COMMAND_LIST]` gesucht wird.
Durch `;` getrennte Befehle werden nacheinander ausgeführt.
Der Text des Schaltflächenetiketts kann mit beliebigem Text nach einem Komma gesetzt werden, das Symbol `\n` fügt einen Zeilenumbruch hinzu.

```
[MDI_COMMAND_LIST]
MDI_COMMAND_MACRO0 = G0 Z25;X0 Y0;Z0, Goto\nUser\nZero
MDI_COMMAND_MACRO1 = G53 G0 Z0;G53 G0 X0 Y0, Goto\nMachn\nZero
```

Action buttons are subclassed from `IndicatedPushButton`. See the following sections for more information about:

- LED Indikator Option
- Enabled on State
- Textänderungen bei Zustand
- Aufruf von Python Befehl zum Zustand (engl. state)

`ActionToolButton` - Optionales Aktionsmenü-Schaltflächen-Widget

`ActionToolButton`-Buttons sind vom Konzept her ähnlich wie Aktions-Buttons, aber sie verwenden *QToolButtons*, um die Auswahl von **optionalen Aktionen** zu ermöglichen, indem man die Taste gedrückt hält, bis das Optionsmenü erscheint.

Derzeit gibt es nur eine Option: `userView`.

Es basiert auf PyQts *QToolButton*.

`userView` *Benutzeransicht Widget aufzeichnen und einstellen*

Die Werkzeugschaltfläche Benutzeransicht ermöglicht das **Aufzeichnen und Zurückkehren zu einer beliebigen Grafikansicht**.

Halten Sie die Taste gedrückt, um das Menü aufzurufen, und drücken Sie *Ansicht aufzeichnen*, um die aktuell angezeigte Grafikansicht aufzuzeichnen.

Klicken Sie normal auf die Schaltfläche, um zur zuletzt aufgezeichneten Position zurückzukehren.

Die aufgezeichnete Position wird beim Herunterfahren gespeichert, wenn eine Einstellungsdateioption

eingerichtet ist.

NOTE Aufgrund von Programmierungseinschränkungen wird die aufgezeichnete Position möglicherweise nicht exakt gleich angezeigt. Dies gilt insbesondere dann, wenn Sie beim Einstellen der gewünschten Ansicht den Ausschnitt verkleinern und wieder verkleinern. Am besten wählen Sie eine Hauptansicht aus, ändern sie wie gewünscht, nehmen sie auf und klicken dann sofort auf die Schaltfläche, um zur aufgezeichneten Position zu wechseln. Wenn sie nicht Ihren Vorstellungen entspricht, ändern Sie die bestehende Position und nehmen Sie erneut auf.

`AxisToolButton` - Achsen-Widget auswählen und einstellen

Damit kann man eine **Achse auswählen und einstellen**.

Wenn die Schaltfläche abhakbar ist, zeigt sie an, welche Achse ausgewählt ist.

Wenn Sie die Taste gedrückt halten, wird ein Popup-Menü angezeigt, in dem Sie folgende Optionen auswählen können:

- Nullen der Achse
- Die Achse durch 2 teilen
- Die Achse beliebig einstellen
- Die Achse auf die zuletzt aufgezeichnete Zahl zurücksetzen

Sie müssen ein Eingabedialog-Widget ausgewählt haben, welcher der Zeichenkette dialog_code_string entspricht, normalerweise wird dieser aus dem screenOptions-Widget ausgewählt.

`halpin_option`

Wird einen HAL Pin auf true setzen, wenn die Achse ausgewählt ist.

`joint_number`

Sollte auf die entsprechende Gelenknummer (engl. joint number) gesetzt werden.

`axis_letter`

Sollte auf den entsprechenden Achsen-Buchstaben gesetzt sein.

Dies sind die Klicke-und-halte (engl. click-and-hold) Menü Eigenschaften:

`showLast`

Zeige die *Setze ans Ende* (set to last) Aktion.

`showDivide`

Zeige die *Teile durch 2* Aktion.

`showGotoOrigin`

Zeige die *Gehe zu G53/G5x Ursprung* Aktion.

showZeroOrigin

Zeige die *Nulle Ursprung* Aktion.

showSetOrigin

Zeige die *Setze Ursprung* Aktion.

dialog_code_string

Legt fest welcher Dialog aufpoppt bei einer numerischen Eingabe, das wäre *ENTRY* (engl. für Eingabe) oder *CALCULATOR* (engl. für Rechner) um einen reinen Eingabedialog oder einen Eingabedialog vom Typ touch/typing calculator aufzurufen.

Hier ist ein Beispiel für einen Stylesheet-Eintrag:

```
AxisToolButton {
    /* Modify all the menu options */
    qproperty-showLast: false;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: false;
    qproperty-dialog_code_string: CALCULATOR;
}
```

Es basiert auf PyQts *QToolButton*.

BasicProbe - Einfaches Fräs-Tast-Widget

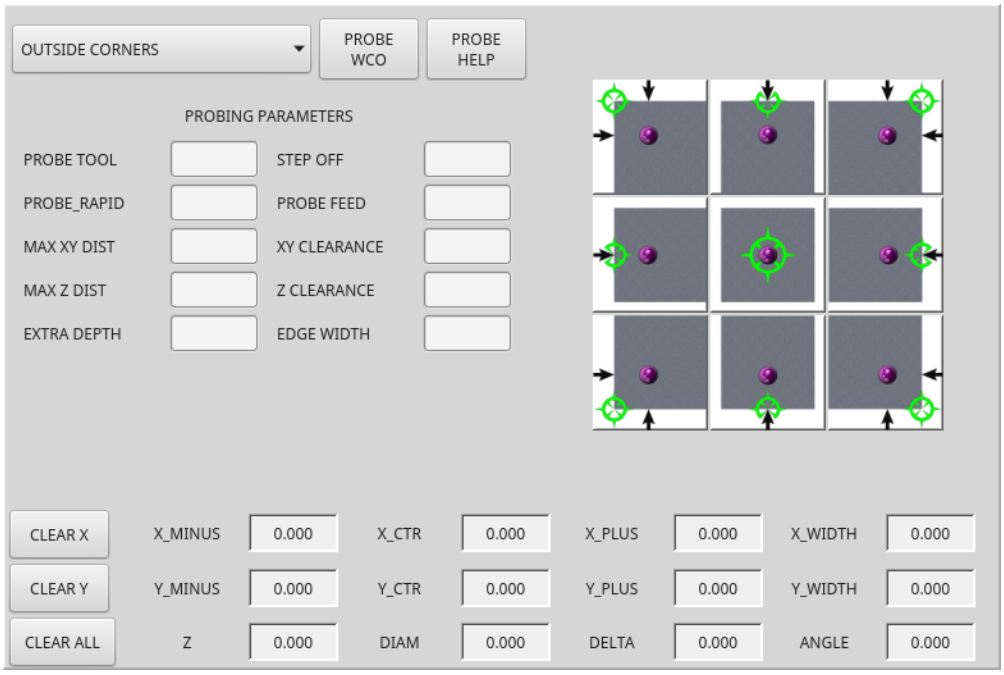


Figure 308. QtVCP BasicProbe: Einfaches Fräs-Tast-Widget

Widget zum **Sondieren auf einer Fräse**. Wird vom *QtDragon*-Bildschirm verwendet.

`CamView` - Widget zur Werkstückausrichtung und Nullpunkteinstellung

Dieses Widget **zeigt ein Bild von einer Webkamera**.

Es *legt ein einstellbares Kreis- und Fadenkreuzziel* über das Bild.

CamView wurde im Hinblick auf eine präzise visuelle Positionierung entwickelt.

Diese Funktion dient der **Ausrichtung des Werkstücks oder der Nullteilmerkmale mithilfe einer Webcam**.

Es verwendet die Vision-Bibliothek *OpenCV*.

`DROLabel` - Widget zur Anzeige der Achsenposition

Damit wird die **aktuelle Position einer Achse angezeigt**.
Sie können auch auf das Label klicken und sehen eine Liste von Aktionen.

`Qjoint_number`

Gelenk Indexnummer (X=0 Y=1) des anzuzeigenden Offsets (10 gibt den Rotationsoffset an).

`Qreference_type`

Tatsächlich, relativ oder noch zu fahrende Entfernung (0,1,2).

`metric_template`

Format der Anzeige, z.B. `%10.3f`.

`imperial_template`

Format der Anzeige, z.B. `%9,4f`.

`angular_template`

Anzeigeformat, z.B. `%Rotational: 10.1f`.

`always_display_diameter`

Wechsel-Schalter der Anzeige (engl. display)-Option

`always_display_radius`

Wechsel-Schalter der Anzeige (engl. display)-Option

`display_as_per_m7m8`

Wechselt Anzeige-Option. Wird dem aktuellen M7/8 Modus folgen.

`follow_reference_changes`

Schaltet die Anzeigeoption ein oder aus. Folgt dem Referenzmodus der STATUS-Nachricht, d. h. Sie können Aktionsschaltflächen verwenden, um festzulegen, wie sie derzeit angezeigt wird.

Dies sind die Click-on-menu Optionen:

showLast

Zeige die *Setze ans Ende* (set to last) Aktion.

showDivide

Zeige die *Teile durch 2* Aktion.

showGotoOrigin

Zeige die *Gehe zu G53/G5x Ursprung* Aktion.

showZeroOrigin

Zeige die *Nulle Ursprung* Aktion.

showSetOrigin

Zeige die *Setze Ursprung* Aktion.

dialogName

Legt fest, welches Dialog-Fenster bei einer numerischen Eingabe erscheint, d.h. ENTRY oder CALCULATOR.

Das `DROLabel`-Widget enthält eine Eigenschaft **`isHomed`**, die mit einem Stylesheet verwendet werden kann, um die _Farbe des `DRO_Label` basierend auf dem Homing-Status der Gelenknummer in LinuxCNC zu ändern.

Hier ist ein Beispiel für einen Stylesheet-Eintrag, der:

- Legt die Schriftart aller *DRO_Label*-Widgets fest,
- Legt die Textvorlage (zur Einstellung der Auflösung) der DRO fest,
- Dann wird die Textfarbe auf der Grundlage der Qt-Eigenschaft `isHomed` eingestellt.
- zeigt alle Menü-Optionen.

```
DROLabel {
    font: 25pt "Lato Heavy";
    qproperty-imperial_template: '%9.4f';
    qproperty-metric_template: '%10.3f';
    qproperty-angular_template: '%11.2f';

    /* Modify all the menu options */
    qproperty-showLast: true;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: true;
    qproperty-dialogName: CALCULATOR;
}

DROLabel[isHomed=false] {
    color: red;
}

DROLabel[isHomed=true] {
```

```
    color: green;
}
```

So geben Sie ein bestimmtes Widget anhand seines `objectName` in Qt Designer an:

```
DROLabel #dr0_x_axis [isHomed=false] {
    color: yellow;
}
```

Es basiert auf PyQts *QLabel*.

`FileManager` - Datei laden Selector Widget (engl. für Dateimanager)

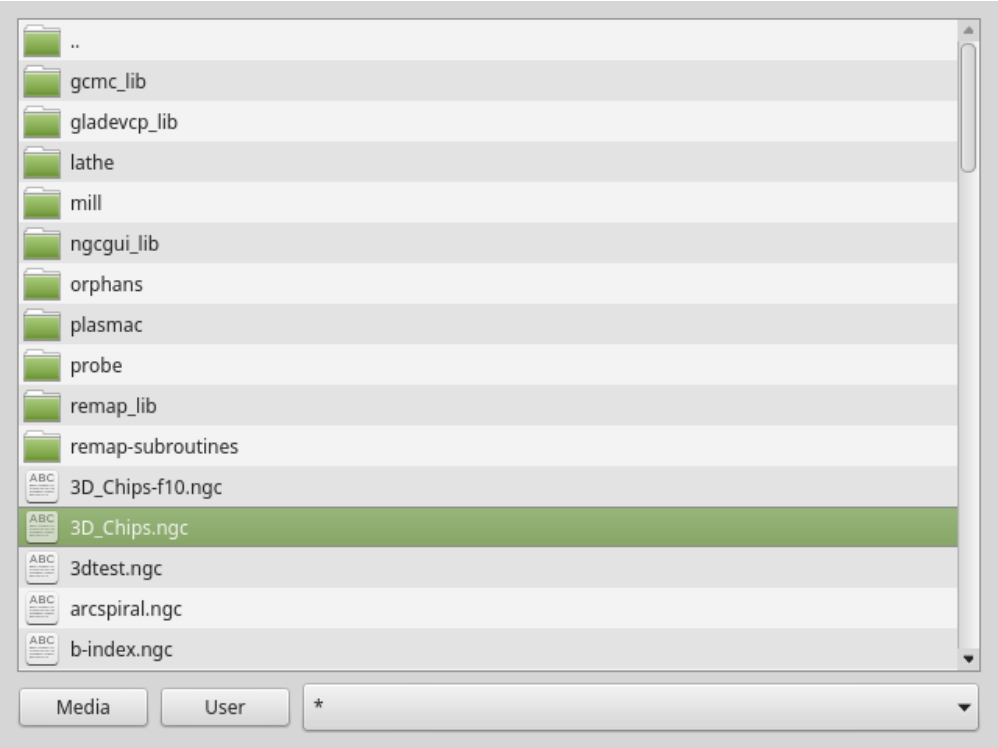


Figure 309. QtVCP `FileManager`: Dateimanager-Widget

Dieses Widget wird verwendet, um **zu ladende Dateien auszuwählen**.

Sie verfügt über die Möglichkeit, die Namen mit Hardware wie einem Handgerät (engl. MPG) zu kennzeichnen.

Man kann die Funktion `load(self,fname)` mit einem Klassenpatch (engl. class patch) versehen, um das Laden von Dateien anzupassen.

Die Funktion `getCurrentSelected()` gibt ein Python-Tupel zurück, das den Dateipfad enthält und angibt, ob es sich um eine Datei handelt.

```
temp = FILEMANAGER.getCurrentSelected()
print('filepath={}'.format(temp[0]))
if temp[1]:
    print('Is a file')
```

Stylesheets Eigenschaften

`doubleClickSelection` *(bool)*

Legt fest, ob ein Doppelklick auf einen Ordner erforderlich ist oder nicht.
Das einfache Anklicken eines Ordners (False) ist standardmäßig aktiviert und ist für Touchscreen-Benutzer gedacht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#filemanager {
    qproperty-doubleClickSelection: True;
}
```

`showListView` *(bool)*

Bestimmt, ob die Datei-/Ordnerstruktur in Listenform angezeigt werden soll oder nicht.
Die Tabellenansicht (False) ist standardmäßig aktiviert.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#filemanager {
    qproperty-showListView: True;
}
```

Es basiert auf PyQt's FIXME

`GcodeDisplay` - G-Code-Textanzeige-Widget

Dies **zeigt G-Code in Textform an**, wobei die aktuell laufende Zeile hervorgehoben wird.

Dies kann auch Folgendes anzeigen:

- **MDI-Verlauf**, wenn sich LinuxCNC im *MDI*-Modus befindet.
- **Log-Einträge**, wenn sich LinuxCNC im *MANUAL*-Modus befindet.
- **Einträge in Präferenzdateien**, wenn Sie `PREFERENCE` in Großbuchstaben in das Widget `MDILine` eingeben.

Es hat ein *Signal* **`percentDone(int)`** (engl. für Prozent erledigt), das mit einem Slot verbunden werden kann (wie z.B. ein `progressBar` (engl. für Fortschrittsbalken), um den Prozentsatz der Ausführung anzuzeigen).

`auto_show_mdi_status`

Setzen Sie true, damit das Widget im MDI-Modus in den MDI-Verlauf wechselt.

`auto_show_manual_status`

Setzen Sie true, damit das Widget im manuellen Modus auf das Maschinenprotokoll umschaltet.

Die `GcodeDisplay`-Eigenschaften können in einem Stylesheet mit folgendem Code eingestellt werden, welcher der .qss-Datei hinzugefügt wird (die folgenden Farbauswahlen sind zufällig).

```
EditorBase{
```

```
    qproperty-styleColorBackground: lightblue;
    qproperty-styleColorCursor:white;
    qproperty-styleColor0: black;
    qproperty-styleColor1: #000000; /* black */
    qproperty-styleColor2: blue;
    qproperty-styleColor3: red;
    qproperty-styleColor4: green;
    qproperty-styleColor5: darkgreen;
    qproperty-styleColor6: darkred;
    qproperty-styleColor7: deeppink;
    qproperty-styleColorMarginText: White;
    qproperty-styleColorMarginBackground: blue;
    qproperty-styleFont0: "Times,12,-1,0,90,0,0,0,0,0";
    qproperty-styleFont1: "Times,18,-1,0,90,1,0,0,0,0";
    qproperty-styleFont2: "Times,12,-1,0,90,0,0,0,0,0";
    qproperty-styleFont3: "Times,12,-1,0,90,0,0,0,0,0";
    qproperty-styleFont4: "Times,12,-1,0,90,0,0,0,0,0";
    qproperty-styleFont5: "Times,12,-1,0,90,0,0,0,0,0";
    qproperty-styleFont6: "Times,12,-1,0,90,0,0,0,0,0";
    qproperty-styleFont7: "Times,12,-1,0,90,0,0,0,0,0";
    qproperty-styleFontMargin: "Times,14,-1,0,90,0,0,0,0,0";
}
```

Für den *Standard-G-Code-Lexer* des Widgets `GcodeDisplay`:

- **styleColor0** = **Standard**: Alles, was nicht zu den folgenden Gruppen gehört
- **styleColor1** = **Zeilennummer und Kommentare**: Nxxx und Kommentare (Zeichen innerhalb und einschließlich *()* oder alles nach *;* (wenn es außerhalb von Klammern verwendet wird) mit Ausnahme der nachstehenden Anmerkung)
- **styleColor2** = **G-code**: G und die nachfolgenden Ziffern
- **styleColor3** = **M-Code**: M und die nachfolgenden Ziffern
- **styleColor4** = **Achse**: XYZABCUVW
- **styleColor5** = **Sonstige**: EFHIJKDQLRPST (Vorschub, Drehzahl, Radius, usw.)
- **styleColor6** = **AchsenWert**: Werte nach XYZABCUVW
- **styleColor7** = **AndererWert**: Werte nach EFHIJKDQLRPST$

NOTE

Für Kommentare kann die Farbe "OtherValue" (Farbe 5) verwendet werden, um "print", "debug", "msg", "logopen", "logappend", "logclose" "log", "pyrun", "pyreload" "abort", "probeopen" "probeclose" innerhalb eines Klammerkommentars in einer G-Code-Zeile. Sowie "py," wenn eine Zeile mit ";py," beginnt. Beispiele: (print, text), (log, text), (msg, text), oder (debug, text). Nur das letzte der Beispiele wird hervorgehoben, wenn es mehrere in derselben Zeile gibt.

Schriftdefinitionen:

```
"style name, size, -1, 0, bold setting (0-99), italics (0-1),
underline (0-1),0,0,0"
```

Es basiert auf PyQts *QsciScintilla*.

`GcodeEditor` - G-Code-Programmeditor-Widget

Dies ist eine Erweiterung des Widgets "GcodeDisplay", welche die Bearbeitung erleichtert.

Es basiert auf PyQt's *QWidget* welches das `GcodeDisplay` Widget beinhaltet.

`GCodeGraphics` - G-Code-Grafik-Backplot-Widget

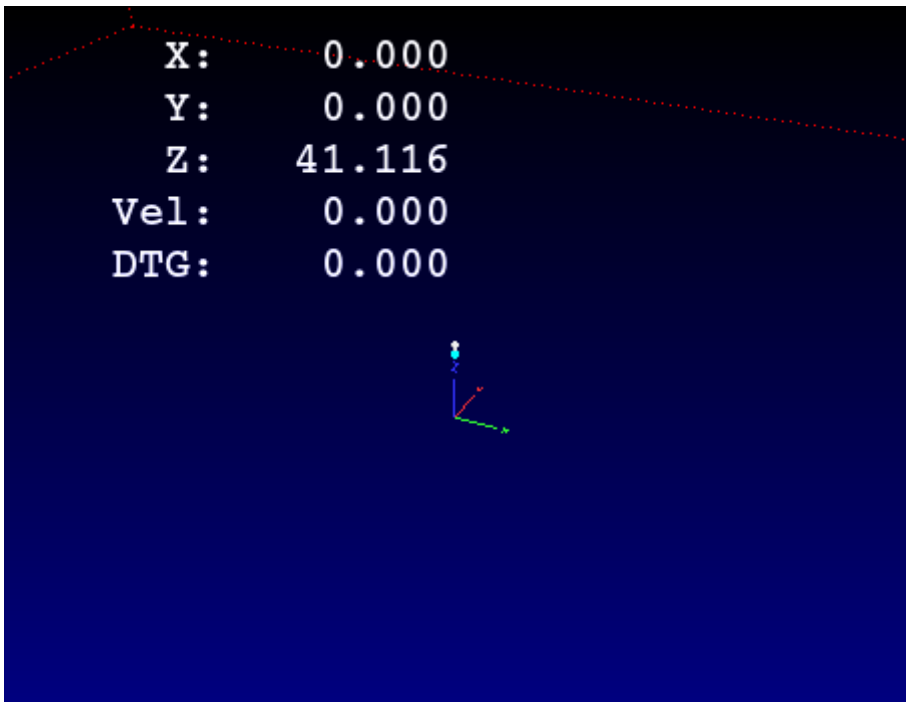


Figure 310. QtVCP GcodeGraphics: G-Code-Grafik-Backplot-Widget

Damit wird die **aktuelle Position einer Achse angezeigt**.

Stylesheets Eigenschaften

`dro-font/dro-large-font` *(string)*

Setzt die kleinen und großen DRO Schrifteigenschaften
Hier referenzieren wir mit dem Widget-Basisnamen; GCodeGraphics

```
GCodeGraphics{
    qproperty-dro_font:"monospace  bold 12";
}
GCodeGraphics{
    qproperty-dro_large_font:"Times 25";
}
```

`_view` *(string)*

Legt die *Standardausrichtung der Ansicht* beim Laden der GUI fest.
Gültige Auswahlmöglichkeiten für eine Drehmaschine sind p, y, y2. Für andere Bildschirme gelten die Optionen p, x, y, z, z2.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt (referenziert über den

durch den widget-Anwender ausgewählten Namen):

```
#gcodegraphics{
    qproperty-_view: z;
}
```

_dro ***(bool)***

Legt fest, ob *die DRO* angezeigt werden soll oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_dro: False;
}
```

_dtg ***(bool)***

Legen Sie fest, ob die *Reststrecke angezeigt* werden soll.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_dtg: False;
}
```

_metric ***(bool)***

Legt fest, ob die_ Einheiten standardmäßig_ in *metrischen* Einheiten angezeigt werden sollen oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_metric: False;
}
```

_overlay ***(bool)***

Legt fest, ob das Overlay standardmäßig angezeigt werden soll oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_overlay: False;
}
```

_offsets ***(bool)***

Legt fest, ob die *Offsets standardmäßig angezeigt* werden sollen oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_offsets: False;
}
```

`_workpiece_opacity` *(float)*

How opaque the faces of *(WORKPIECE,...)* stock are drawn, from 0 - the wireframe outline alone - to 1. The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_workpiece_opacity: 0.25;
}
```

`_small_origin` *(bool)*

Legt fest, ob *standardmäßig der kleine Ursprung angezeigt wird.*
Das folgende Beispiel zeigt, wie diese Eigenschaft festgelegt wird:

```
#gcodegraphics{
    qproperty-_small_origin: False;
}
```

`overlay_color` *(Primär-, Sekundär- oder RGBA-formatierte Farbe)*

Legt die *Standard-Overlayfarbe* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-overlay_color: blue;
}
```

`overlay_alpha` *(float)*

Legt die *Standard-Overlay Durchsichtigkeit* (engl. alpha value) fest. Dies beeinflusst die Opazität des Overlays, wenn zwischen 0,0 und 1,0 eingestellt.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-overlay_alpha: 0.15;
}
```

`background_color ` Hintergrundfarbe, *(Primär-, Sekundär- oder RGBA-formatierte Farbe)*

Legt die *Standard-Hintergrundfarbe* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-background_color: blue;
}
```

`+_use_gradient_background+` *(bool)*

Legt fest, ob *Standardmäßig ein Hintergrund mit Farbverlauf verwendet wird.*
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_use_gradient_background: False;
```

```
}
```

`jog_color` ***(Primär-, Sekundär- oder RGBA-formatierte Farbe)***

Legt die *Standard-Jog-Farbe* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-jog_color: red;
}
```

`Feed_color` ***(Primär-, Sekundär- oder RGBA-formatierte Farbe)***

Legt die *Standard-Farbe für den Vorschub* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-Feed_color: green;
}
```

`Rapid_color` ***(Primär-, Sekundär- oder RGBA-formatierte Farbe)***

Legt die *Standard-Farbe für den Eilgang* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-Rapid_color: rgba(0, 0, 255, .5);
}
```

`InhibitControls` ***(bool)***

Legt fest, ob *externe Steuerelemente standardmäßig gesperrt* werden sollen oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-InhibitControls:True;
}
```

`MouseButtonMode` ***(int)***

Ändert das Verhalten der *Maustaste* zum Drehen, Verschieben oder Zoomen innerhalb der Vorschau.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-MouseButtonMode: 1;
}
```

Es gibt 12 gültige Modi:

Mode	**Move**	**Zoom**	**Rotate** (engl. drehen)

0	Links	Mitte	Rechts
1	Mitte	Rechts	Links
2	Mitte	Links	Rechts
3	Links	Rechts	Mitte
4	Rechts	Links	Mitte
5	Rechts	Mitte	Links

Die Modi 6-11 sind für Maschinen gedacht, die nur eine 2D-Vorschau benötigen, wie z. B. Plasmageräte oder einige Drehmaschinen, denen keine Drehtaste zugewiesen ist.

Mode	**Move**	**Zoom**
6	Links	Mitte
7	Mitte	Links
8	Rechts	Links
9	Links	Rechts
10	Mitte	Rechts
11	Rechts	Mitte

`MouseWheelInvertZoom` *(bool)*

Legt fest, ob beim Zoomen mit dem Mausrad die *Zoomrichtung invertiert* werden soll oder nicht. Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-MouseWheelInvertZoom:True;
}
```

`ACTION` *Funktionen*

Die Bibliothek `ACTION` kann das G-Code-Grafik-Widget steuern.

`ACTION.RELOAD_DISPLAY()`

Lädt das aktuelle Programm neu, das den Ursprung/die Offsets neu berechnet.

`ACTION.SET_GRAPHICS_VIEW(_view_)`

Folgende `view`-Befehle können gesendet werden:

- `clear`
- `zoom-in`
- `zoom-out`
- `pan-up`
- `pan-down`

- pan-right
- pan-left
- rotate-cw
- rotate-ccw
- rotate-up
- rotate-down
- overlay-dro-on
- overlay-dro-off
- overlay-offsets-on
- overlay-offsets-off
- alpha-mode-on
- alpha-mode-off
- inhibit-selection-on
- inhibit-selection-off
- dimensions-on
- dimensions-off
- grid-size
- record-view
- set-recorded-view
- P
- X
- Y
- Y2
- Z
- Z2
- *set-large-dro*
- *set-small-dro*

ACTION.ADJUST_PAN(_X,Y_)

Legen Sie direkt den relativen Blickwinkel in x- und y-Richtung fest.

ACTION.ADJUST_ROTATE(_X,Y_)

Legen Sie direkt die relative Drehung der Ansicht in x- und y-Richtung fest.

Es basiert auf dem *OpenGL* Widget von PyQt.

`JointEnableWidget` - FIXME

FIXME JointEnableWidget Dokumentation

`JogIncrements` - Auswahl-Widget für Jog-Inkremente

Mit diesem Widget kann der Benutzer **Werte für die Schrittweite beim Joggen auswählen**.

Die Joggingwerte stammen aus der *INI-Datei* unter:

- `[DISPLAY]INCREMENTS`, or
- `[DISPLAY]ANGULAR_INCREMENTS`

Dies wird *für alle Widgets* über `STATUS` verfügbar sein.

Sie können lineare oder Winkelinkremente durch die Eigenschaft **`linear_option`** im Eigenschafteneditor von Qt Designer auswählen.

Es basiert auf PyQts *ComboBox*.

`MacroTab` - Spezielles Makro-Widget



Figure 311. QtVCP `MacroTab`: Spezielles Makro-Widget

Mit diesem Widget kann der Benutzer **spezielle Makroprogramme** für die Erledigung kleinerer Aufgaben auswählen und anpassen.

Es verwendet *Bilder zur visuellen Darstellung* des Makros und für ein Symbol.

Es sucht nach speziellen Makros unter Verwendung der *INI-Definition*:

```
[RS274NGC]
SUBROUTINE_PATH =
```

Die Makros sind **`O-Wort`-Unterprogramme mit speziellen Kommentaren** für die Zusammenarbeit mit dem Launcher. Die ersten drei Zeilen *müssen* die untenstehenden Schlüsselwörter enthalten, die vierte ist optional.

Hier ist ein Beispiel für die ersten vier Zeilen einer *O-Word-Datei*:

```
; MACROCOMMAND = Entry1,Entry2
; MACRODEFAULTS = 0,true
; MACROIMAGE = my_image.svg,Icon layer number,Macro layer number
; MACROOPTIONS = load:yes,save:yes,default:default.txt,path:~/macros
```

MACROCOMMAND

Dies ist die *erste Zeile* in der O-Wort-Datei.

Es handelt sich um eine **durch Kommata getrennte Liste von Text, der über einem Eintrag angezeigt werden soll**.
Es gibt **eine für jede erforderliche Variable** in der O-Wort-Funktion.

Wenn das Makro keine Variablen benötigt, lassen Sie es leer:

```
; MACROCOMMAND=
```

MACRODEFAULTS

Dies muss die *zweite Zeile* in der O-Wort-Datei sein.

Es handelt sich um eine **durch Kommata getrennte Liste der Standardwerte für jede Variable** in der O-Wort-Funktion.

Wenn Sie das Wort `radiotrue`, `radiofalse`, `true` oder `false` in der Liste verwenden, wird ein `*radiobutton*' angezeigt. + Wenn Sie das Wort `checktrue` oder `checkfalse` in der Liste verwenden, wird eine `*checkbox*` angezeigt.
Wenn Sie das Wort `buttontrue` oder `buttonfalse` in der Liste verwenden, wird ein `**Checkable Pushbutton** ` angezeigt.
Wenn die Voreinstellung einen Dezimalpunkt hat, geht Makro Tab davon aus, dass Sie eine Gleitkommazahl wünschen, ansonsten eine ganze Zahl.

NOTE	Wenn Sie Radiobuttons nutzen, setzen Sie nur einen Radiobutton auf true. Radiobuttons werden für sich gegenseitig ausschließende Optionen genutzt.

MACROIMAGE

Dies muss die *dritte Zeile* in der O-Wort-Datei sein.

- **SVG-Bilder**
  Wenn Sie SVG-Bilddateien verwenden, müssen sie mit der Erweiterung `.svg` enden.

  Die Bilder müssen zu *SVG-Ebenen* hinzugefügt werden, die zur Definition der verschiedenen Bilder für Makro und Symbol verwendet werden.

Wert ist eine durch Kommata getrennte Liste von drei geordneten Feldern:

```
; MACROIMAGE=filename.svg,macro_layer_name[,icon_layer_name]
```

Mit:

dateiname.svg

Name der SVG-Bilddatei als erstes Feld.
Es wird davon ausgegangen, dass sie sich im selben Ordner befindet wie die O-Wort-Datei.

***macro_layer_name**

Name der Makrobildebene als zweites Feld.

icon_layer_name

Name der Ikonenebene als optionales drittes Feld. Fehlt der dritte Eintrag, wird für Makro und Symbol das gleiche Bild verwendet.

- **PNG/JPG-Bilder**:
  Wert bleibt eine durch Komma getrennte Liste:

```
; MACROIMAGE=macro_image.(png|jpg)[,icon_image.(png|jpg)]
```

Mit:

_macro_image_.(png|jpg)

Name der Bilddatei des Makros als erstes Feld.
Es wird davon ausgegangen, dass sich die Bilddatei im selben Ordner befindet wie das Makro.

_icon_image_.(png|jpg)

Bild-Dateiname des Icons als optionales zweites Feld.
Fehlt der zweite Eintrag, wird für Makro und Bild das gleiche Bild verwendet und Bild verwendet.

Wenn das Schlüsselwort vorhanden ist, aber die Einträge fehlen, werden keine Bilder verwendet.

MACROOPTIONS

Diese *optionale Zeile muss die vierte* Zeile in der O-Wort-Datei sein.

Es handelt sich um eine durch Kommata getrennte Liste von Schlüsselwörtern und Daten, davon sind alle optional:

LOAD:yes

Zeigt einen Button zum Laden an.

SAVE:yes

Zeigt einen Button zum Speichern an.

DEFAULT:ThisMacroData.txt

Legt den standardmäßig vorausgewählten Dateinamen fest, wenn Daten für dieses Makro geladen

oder gespeichert werden.
Der Name kann beliebig gültig sein, muss jedoch auf *.txt* enden.

PATH:~/linuxcnc/nc_files/mySavedMacrosData

Setzt das voreingestellte Verzeichnis zur Vorauswahl wenn Daten für dieses Makro geladen oder gespeichert werden.

MacroTab Stylesheets

Hier sind Stylesheet-Hinweise zum Anpassen des MacroTab-Widgets.

```
MacroTab CustomButton{
    width: 20px;
    height: 40px;
}

MacroTab QPushButton {
    width: 80px;
    height: 40px;
}

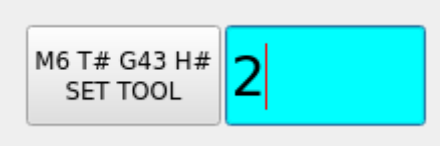
MacroTab QLabel {
    font: 24pt "Lato Heavy";
}

TouchSpinBox  LineEdit {
    font: 12pt "Lato Heavy";
}

TouchSpinBox  QPushButton {
    width: 60px;
    height: 100px;
}
```

`OperatorValueLine` - Operator Value Line Eingabe-Widget

Der Bediener gibt Werte in dieses Widget ein, die auf eine Vorlage angewendet und optional entweder sofort an das MDI gesendet oder zu einem späteren Zeitpunkt verwendet werden. Das Widget unterstützt die optionale Einblendung eines Taschenrechners, einer Tastatur oder eines Werkzeugauswahldialogs für eine touchfreundliche Eingabe, indem die `dialog_keyboard_option` gesetzt wird. Um zu ändern, welcher Dialogtyp angezeigt wird, bearbeiten Sie die `dialog_code_option`.



Formatting MDI Befehl

Das Widget unterstützt eine Formatierungsoption, die an Pythons String-Methode `format()` übergeben

wird, um die endgültige Ausgabe für den MDI-Befehl zu erzeugen. Das spezielle Token `{value}` kann an beliebiger Stelle in dieser Formatzeichenkette eingefügt werden, an welcher der Wert erscheinen soll. Die Formatierungseigenschaft heißt `mdi_command_format_option`, z. B.:

- `M3 S{value}` startet die Spindel mit der vom Bediener eingegebenen Drehzahl.
- `M6 T{value} G43 H{value}` führt einen Werkzeugwechsel sowie eine Werkzeuglängenkompensation basierend auf der eingegebenen Werkzeugnummer aus.

Automatisches vs. verzögertes (deferred) MDI-Problem

Das Widget kann so konfiguriert werden, dass es den MDI-Befehl beim Absenden automatisch ausführt, wenn `issue_mdi_on_submit_option` auf `True` gesetzt ist. Ist der Wert `False`, kann die Ausführung des Befehls zu einem späteren Zeitpunkt über ein Signal oder einen Funktionsaufruf von einem anderen Widget erfolgen.

In Fällen, in denen `issue_mdi_on_submit_option` auf `False` gesetzt ist, führt ein Aufruf der Funktion `issue_mdi()` den Befehl aus. Slots, die mit Widgets wie PushButtons verknüpft sind, können den MDI-Befehl beim Drücken auslösen, z. B.:

```
    def setSpindleSpeed(self, event):
        self.w.lineSpindleSpeed.issue_mdi()
        ACTION.SET_MANUAL_MODE()

    def setToolNumber(self, event):
        self.w.lineToolNumber.issue_mdi()
        ACTION.SET_MANUAL_MODE()
```

"Noch auszuführen" (engl. pending) Zustand Styling Beispiel

Das Widget verfolgt über die Eigenschaft `isPendingValue`, ob ein eingegebener Wert noch aussteht und noch nicht ausgeführt wurde. Diese Eigenschaft kann zur Gestaltung des Widgets über das Stylesheet verwendet werden. So lässt sich der Bediener darauf hinweisen, dass zwar ein Wert eingegeben wurde, jedoch eine weitere Aktion erforderlich ist, um ihn anzuwenden.

Im folgenden Stylesheet-Auszug wird das Eingabe-Widget mit einem Cyan-Hintergrund hervorgehoben, wenn Werte anhängig (engl. pendig) sind und bislang nicht angewendet wurden.

```
#lineSpindleSpeed[isPendingValue=true],
#lineToolNumber[isPendingValue=true] {
    background: cyan;
}

#lineSpindleSpeed[isPendingValue=false],
#lineToolNumber[isPendingValue=false] {
    background: none;
}
```

`MDILine` - MDI-Befehlszeileneingabe-Widget

Hier kann man **MDI-Befehle** eingeben.

Eine Popup-Tastatur ist verfügbar.

Eingebettete Befehle

Es gibt auch **eingebettete Befehle**, die über dieses Widget verfügbar sind.

Geben Sie einen dieser Befehle ein, um das entsprechende Programm zu laden oder die Funktion aufzurufen:

`HALMETER`

Startet das LinuxCNC `halmeter` Dienstprogramm.

`HALSHOW`

Starts LinuxCNC `halshow` utility.

`HALSCOPE`

Startet das LinuxCNC `halscope` Dienstprogramm.

`STATUS`

Startet das LinuxCNC `status` Dienstprogramm.

`CALIBRATION`

Startet die LinuxCNC Calibration

`CLASSICLADDER`

Startet die ClassicLadder GUI, wenn die *ClassicLadder realtime HAL component* durch die Konfigurationsdateien des Rechners geladen wurde.

`PREFERENCE`

Lädt die Einstellungsdatei in den `GcodeEditor`.

`CLEAR HISTORY`

Löscht den MDI-Verlauf.

`net`

See `halcmd net` commands.
An error will result if the command is unsuccessful.

- *Syntax*: `net <signal name> <pin name>`
- *Beispiel*: `net plasmac:jog-inhibit motion.jog-stop`

`setp`

Sets den Wert eines Pins oder einer parameter.
Gültige Werte hängen vom Objekttyp des Pins oder Parameters ab.
Dies führt zu einem Fehler, wenn die Datentypen nicht übereinstimmen oder der Pin mit einem Signal verbunden ist.

- *Syntax*: `setp <Pin/Parameter-Name> <Wert>`
- *Beispiel*: `setp plasmac.resolution 100`

`unlinkp`

Trennt einen Pin von einem Signal.
Ein Fehler tritt auf, wenn der Pin nicht vorhanden ist.
Running LinuxCNC von Terminal kann helfen, die Ursache zu bestimmen, wie Fehlermeldungen von `hal_lib.c` wird dort angezeigt werden.

- *Syntax*: `unlinkp <Pin-Name>`
- *Beispiel*: `unlinkp motion.jog-stop`

NOTE	Die Funktion `MDILine` **`spindle_inhibit`** kann von der Handler-Datei einer grafischen Benutzeroberfläche verwendet werden, um die Spindelbefehle `M3`, `M4` und `M5` bei Bedarf zu sperren.

Es basiert auf PyQts *QLineEdit*.

`MDIHistory` - MDI-Befehlsverlaufs-Widget

Zeigt eine **scrollbare Liste vergangener MDI-Befehle** an.

Für MDI-Befehle wird eine Bearbeitungszeile eingebettet. Auf die gleichen eingebetteten MDILine-Befehle kann von diesem Widget aus zugegriffen werden.

Der Verlauf wird *in einer Datei aufgezeichnet, die in der INI* unter der Überschrift `[DISPLAY]` definiert ist (dies ist die Standardeinstellung):

```
MDI_HISTORY_FILE = '~/.axis_mdi_history'
```

`MDITouchy` - Touchscreen-MDI-Eingabe-Widget

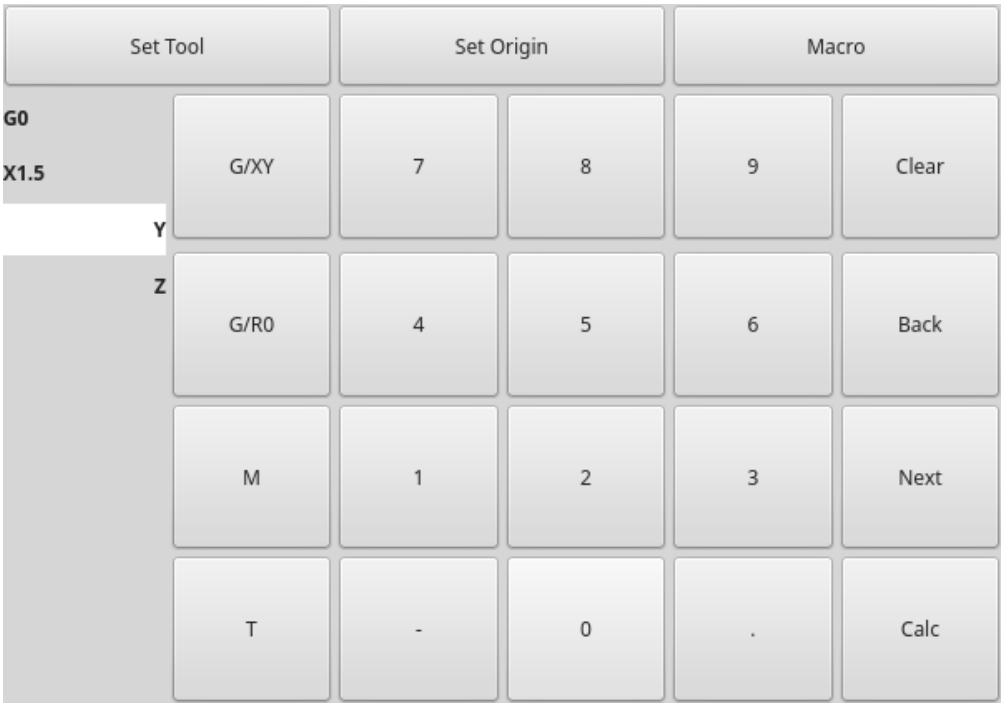


Figure 312. QtVCP `MDITouchy`: Touchscreen-MDI-Eingabe-Widget

Dieses Widget zeigt **Buttons und Eingabezeilen für die Eingabe von MDI-Befehlen** an.

Basierend auf LinuxCNC's Touchy Screen's MDI-Eingabe-Prozess, dessen großen Buttons sind sehr nützlich für Touchscreens.

So verwenden Sie `MDITouchy`:

- Drücken Sie zunächst eine der Tasten `G/XY`, `G/R0`, `M` oder `T`. Auf der linken Seite werden die Eingabefelder angezeigt, die ausgefüllt werden können.
- Drücken Sie dann „Weiter" und „Zurück", um zwischen den Feldern zu navigieren.
- `Calc` öffnet einen Taschenrechnerdialog.
- `Clear` löscht den aktuellen Eintrag.
- `Set Tool` wird einen Werkzeugwechsel fordern.
- `Set Origin` ermöglicht das Festlegen des Ursprungs des aktuellen G6x-Systems.
- `Macro` ruft alle verfügbaren Makro-ngc-Programme auf.

Das Widget *erfordert einen expliziten Aufruf des MDITouchy-Python-Codes*, um den *MDI-Befehl tatsächlich auszuführen*:

- **Für Handler-Datei-Code**
  Wenn das Widget in Qt Designer *mditouchy* genannt wurde, würde der folgende Befehl den angezeigten MDI-Befehl ausführen:

  ```
  self.w.mditouchy.run_command()
  ```

- **Für die Verwendung der Aktionstaste**
  Wenn das Widget in Qt Designer "mditouchy" genannt wurde, verwenden Sie die Option "Python-

Befehle aufrufen" der Aktionsschaltfläche und geben Sie Folgendes ein:

```
INSTANCE.mditouchy.run_command()
```

Die Makro-Schaltfläche *durchläuft die in der INI-Überschrift [ANZEIGE] definierten Makros*.

Fügen Sie eine oder mehrere MACRO-Zeilen im folgenden Format hinzu:

```
MACRO = macro_name [param1] [... paramN]
```

Im folgenden Beispiel ist `increment` der Name des Makros, und es akzeptiert zwei Parameter, die `xinc` und `yinc` heißen.

```
MACRO = increment xinc yinc
```

Legen Sie nun das Makro in einer Datei mit dem Namen `macro_name.ngc` im Verzeichnis `PROGRAM_PREFIX` oder in einem beliebigen Verzeichnis im `SUBROUTINE_PATH` ab, das in der INI-Datei angegeben ist.

Um bei dem obigen Beispiel zu bleiben, würde es `increment.ngc` heißen und sein Inhalt könnte wie folgt aussehen:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Beachten Sie, dass der *Name des Unterprogramms exakt mit dem Dateinamen und dem Makronamen übereinstimmt*, einschließlich Groß- und Kleinschreibung.

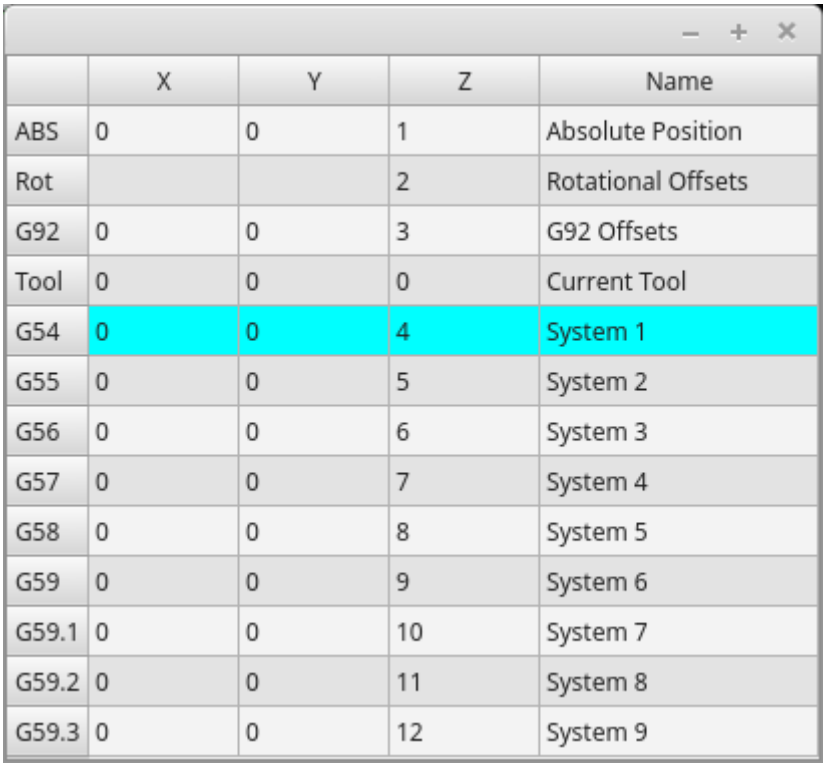
Wenn Sie das Makro durch Drücken der Schaltfläche Makro aufrufen, können Sie Werte für Parameter eingeben (in unserem Beispiel "xinc" und "yinc").
Diese werden als Positionsparameter an das Makro übergeben: `#1`, `#2`... `#N` jeweils.
Parameter, die Sie leer lassen, werden als Wert `0` übergeben.

Wenn es mehrere verschiedene Makros gibt, drücken Sie wiederholt die Makrotaste, um sie zu durchlaufen.

Wenn Sie in diesem einfachen Beispiel -1 für xinc eingeben und die Ausführung des MDI-Zyklus aufrufen, wird eine schnelle *G0*-Bewegung ausgelöst, die eine Einheit nach links geht.

Diese Makrofunktion ist nützlich für das Antasten von Kanten/Löchern und andere Einrichtungsaufgaben sowie vielleicht für das Fräsen von Löchern oder andere einfache Operationen, die vom Bedienfeld aus durchgeführt werden können, ohne dass speziell geschriebene G-Code-Programme erforderlich sind.

`OriginOffsetView` - Ursprungsansicht und Einstellungs-Widget



	X	Y	Z	Name
ABS	0	0	1	Absolute Position
Rot			2	Rotational Offsets
G92	0	0	3	G92 Offsets
Tool	0	0	0	Current Tool
G54	0	0	4	System 1
G55	0	0	5	System 2
G56	0	0	6	System 3
G57	0	0	7	System 4
G58	0	0	8	System 5
G59	0	0	9	System 6
G59.1	0	0	10	System 7
G59.2	0	0	11	System 8
G59.3	0	0	12	System 9

Figure 313. QtVCP `OriginOffsetsView`: Origins-Ansicht und Einstellungs-Widget

Dieses Widget ermöglicht es, **Offsets von Benutzer-spezifizierten Ursprüngen** direkt zu **visualisieren und zu ändern**.

Es wird *die Parameterdatei von LinuxCNC* für vorgenommene oder gefundene Änderungen aktualisieren.

Die Einstellungen können in LinuxCNC nur nach der Referenzfahrt und im Ruhezustand des Motion Controllers geändert werden.

Die Anzeige und Eingabe wird zwischen metrischen und imperialen Maßeinheiten, basierend auf LinuxCNC's *aktuellen* `G20` / `G21` Einstellung ändern.

Das aktuell genutzte Benutzersystem wird hervorgehoben.

Es können zusätzliche Aktionen integriert werden, um Einstellungen zu manipulieren.
Diese Aktionen hängen von zusätzlichem Code ab, der entweder zu einem kombinierten Widget, wie dem `originoffsetview`-Dialog, oder dem Bildschirm-Handler-Code hinzugefügt wird.
Typische Aktionen sind z.B. *Clear Current User offsets* oder *Zero X*.

Wenn Sie auf die Spalten und Zeilen klicken, können Sie die Einstellungen anpassen.

Für die Daten- oder Texteingabe kann ein Dialogfeld eingeblendet werden.

Der Abschnitt mit den Kommentaren wird in die Einstellungsdatei aufgenommen.

Es basiert auf PyQt's *QTableView*, *QAbstractTableModel*, und *ItemEditorFactory*.
Eigenschaften, Funktionen und Stile der PyQt-Basisobjekte sind immer verfügbar.

Eigenschaften

`OriginOffsetView` hat die folgenden Eigenschaften:

`dialog_code_string`

Legt fest, welcher Dialog bei einer numerischen Eingabe erscheint.

`test_dialog_code_string`

Legt fest, welcher Dialog bei der Texteingabe angezeigt wird.

`metric_template`

Metrisches numerisches Datenformat.

`imperial_template`

Imperiales numerisches Datenformat.

`styleCodeHighlight`

Aktuell verwendete Farbe für die Hervorhebung des Benutzersystems.

Diese können eingestellt werden in:

- Qt Designer, in
- Python-Handler-Code

```
self.w.originoffsetview.setProperty('dialog_code','CALCULATOR')
self.w.originoffsetview.setProperty('metric_template','%10.3f')
```

- Oder (gegebenenfalls) in Stylesheets

```
OriginOffsetView{
  qproperty-styleColorHighlist: lightblue;
}
```

`RadioAxisSelector` - FIXME

FIXME RadioAxisSelector-Dokumentation

`RoundButton` - Rundes `ActionButton` Widget

Runde Schaltflächen funktionieren genauso wie *ActionButtons* mit dem Unterschied, dass die Schaltfläche rund abgeschnitten ist.

Sie sollen sich nur optisch unterscheiden.

Sie haben *zwei Pfadeigenschaften* für die Anzeige von **Bildern auf true und false**.

StateLabel - Controller-Modi Statusbeschriftungsanzeige-Widget

Dadurch wird eine **Beschriftung basierend auf den Zuständen der Maschinensteuerung true/false** angezeigt.

Sie können zwischen verschiedenen Texten wählen, die auf wahr oder falsch basieren.

Eigenschaften der Zustandsauswahl

Die Zustände sind über diese Eigenschaften wählbar:

`css_mode_status`

True, wenn sich die Maschine in *G96* befindet *Constant Surface Speed Mode*.

`diameter_mode_status`

True, wenn sich die Maschine in *G7* befindet *Drehmaschine Durchmesser Modus*.

`fpr_mode_status`

`True`, wenn die Maschine im `G95` *Vorschub je Umdrehung Modus* (engl. Feed per Revolution Mode) ist.

`metric_mode_status`

`True`, wenn sich die Maschine im `G21` *Metrischen Modus* befindet.

Eigenschaften von Textvorlagen

`true_textTemplate`

Dies ist der Text, der gesetzt wird, wenn die Option `True` ist.
Sie können *Qt rich text* code für verschiedene Schriftarten/Farben usw. verwenden.
Eine typische Vorlage für den metrischen Modus im wahren Zustand könnte sein: *Metrischer Modus*

`false_textTemplate`

Dies ist der Text, der gesetzt wird, wenn die Option `False` ist.
Sie können *Qt rich text* code für verschiedene Schriftarten/Farben usw. verwenden.
Eine typische Vorlage für den metrischen Modus im falschen Zustand könnte sein: *Imperialer Modus*.

Es basiert auf PyQts *QLabel*.

`StatusLabel` - Anzeige-Widget für Controller-Variablen-Zustandsbeschriftung

Dadurch wird ein Etikett angezeigt, das auf dem auswählbaren Status der Maschinensteuerung basiert. Sie können ändern, wie der Status angezeigt wird, indem Sie den Python-Formatierungscode in der Textvorlage ersetzen. Sie können auch Rich Text für verschiedene Schriftarten/Farben usw. verwenden.

Auswählbare Zustände

Diese Zustände sind wählbar:

`actual_spindle_speed_status`

Dient zur Anzeige der tatsächlichen Spindeldrehzahl, wie sie vom HAL-Pin "spindle.0.speed-i" gemeldet wird.
Sie wird in *RPM* umgewandelt.

Normalerweise wird ein "textTemplate" von "%d" verwendet.

actual_surface_speed_status

Dient zur Anzeige der tatsächlichen Schnittflächengeschwindigkeit auf einer Drehmaschine basierend auf der X-Achse und der Spindeldrehzahl.
Sie wird in Entfernung pro Minute umgerechnet.
Normalerweise wird ein "textTemplate" von "%4.1f" (Fuß pro Minute) und "altTextTemplate" von "%d" (Meter pro Minute) verwendet.

blendcode_status

Zeigt die aktuelle G64-Einstellung.

current_feedrate_status

Zeigt die Vorschubgeschwindigkeit.

current_FPU_status

Zeigt den aktuellen tatsächlichen Vorschub pro Einheit an.

fcode_status

Zeigt die aktuelle programmierte F-Code-Einstellung an.

feed_override_status

Zeigt die aktuelle Einstellung des Vorschub-Overrides in Prozent an.

filename_status

Zeigt den Namen der zuletzt geladenen Datei an.

filepath_status

Zeigt den Namen des zuletzt geladenen vollständigen Dateipfad an.

gcode_status

Zeigt alle aktiven G-Codes.

gcode_selected_status

Zeigt die aktuell ausgewählte G-Code-Zeile an.

halpin_status

Zeigt den HAL-Pin-Ausgang eines ausgewählten HAL-Pin.

jograte_status

Zeigt die aktuelle QtVCP-basierte Jog-Rate an.

jograte_angular_status

Zeigt die aktuelle QtVCP-basierte Winkeljograte an.

jogincr_status

Zeigt den aktuellen QtVCP-basierten Jog-Inkrement an.

jogincr_angular_status

Zeigt den aktuelle QtVCP-basierte Winkel Jog-Inkrement an.

machine_state_status

Zeigt den aktuellen *Maschineninterpreter-Zustand* unter Verwendung des aus der Maschinen-Zustandsliste (state_list) beschriebenen Textes an.
Die Interpreter-Zustände sind:

- `Estopped`
- `Running`
- `Stopped`
- `Paused`
- `Waiting`
- `Reading`

max_velocity_override_status

Zeigt die aktuelle Einstellung für die maximale Achsengeschwindigkeit an.

mcode_status

Zeigt *alle aktiven M-Codes*.

motion_type_status

Zeigt die aktuelle Art der Maschinenbewegung unter Verwendung des in der motion_type_list beschriebenen Textes an.

- *None*
- *Rapid* (engl. für Schnellauf)
- *Feed* (engl. für Vorschub)
- *Arc* (engl. für Bogen)
- *Tool Change* (engl. für Werkzeugwechsel)
- *Probe* (engl. für Sonde)
- *Rotary Index* (engl. für Rotationsindex)

requested_spindle_speed_status

Zeigt die gewünschte Spindeldrehzahl an - die tatsächliche kann abweichen.

rapid_override_status

Zeigt die aktuelle Einstellung des Vorschub-Overrides in Prozent an.

spindle_override_status

Zeigt die aktuelle Spindel-Override-Einstellung in Prozent an.

timestamp_status

Zeigt die Zeit auf der Grundlage der Systemeinstellungen an.

Ein Beispiel für eine nützliche `textTemplate`-Einstellung: `%I:%M:%S %p`.
Siehe das Python-Zeitmodul für weitere Informationen.

`tool comment_status`

Gibt den Kommentartext des aktuell geladenen Werkzeugs zurück.

`tool diameter_status`

Gibt den Durchmesser des aktuell geladenen Werkzeugs zurück.

`tool_number_status`

Gibt die Werkzeugnummer des aktuell geladenen Werkzeugs zurück.

`tool_offset_status`

Gibt den Offset des aktuell geladenen Werkzeugs zurück, indiziert durch `index_number` zur Auswahl der Achse (0=x,1=y, u.s.w.).

`user_system_status`

Zeigt das *aktive Benutzerkoordinatensystem* (Einstellung `G5x`).

Weitere Eigenschaften

`index_number`

Integerwert, der den anzuzeigenden Werkzeugstatus-Index angibt.

`state_label_list`

Liste der für die verschiedenen Maschinenzustände verwendeten Bezeichnungen.

motion_label_list

Liste der für die verschiedenen Bewegungsarten verwendeten Bezeichnungen.

`halpin_names`

Name eines zu überwachenden Halpins (einschließlich des Basisnamens der HAL-Komponente).

`textTemplate`

Dies wird normalerweise für **imperiale (`G20`) oder eckige numerische Einstellungen** verwendet, obwohl nicht jede Option eine imperiale/metrische Umrechnung hat.
Dies verwendet *Python-Formatierungsregeln*, um die Textausgabe festzulegen.
Man kann `%s` für keine Konvertierung, `%d` für Integer-Konvertierung, `%f` für Float-Konvertierung, etc. verwenden.
Sie können auch *Qt Rich Text* Code verwenden.
Eine typische Vorlage für die Formatierung von imperialen Fließkommazahlen in Text wäre `%9.4f` oder `%9.4f inch`.

`alt_textTemplate`

Dies wird normalerweise für **metrische (`G21`) numerische Einstellungen** verwendet.
Dies verwendet *Python Formatierungsregeln*, um die Textausgabe festzulegen.
Eine typische Vorlage für die Formatierung von metrischen Fließkommazahlen in Text wäre "%10.3f" oder "%10.3f mm".

Es basiert auf PyQts *QLabel*.

`StatusImageSwitcher` - Controller-Statusbildumschalter

Der Status-Image-Switcher **wechselt zwischen Images, die auf LinuxCNC-Zuständen basieren**.

*`watch_spindle`

Wechselt zwischen *3 Bildern*: `stop`, `fwd`, `revs`.

*`watch_axis_homed`

Watch Axis Homed würde zwischen 2 Bildern umschalten: `axis not homed` (engl. für Achse nicht referenziert), `axis homed` (Achse referenziert).

*`watch_all_homed`

Würde zwischen *2 Bildern* wechseln: `not all homed` (nicht alle referenzert), `all homed` (alle referenziert).

*`watch_hard_limits`

Würde zwischen *2 Bildern oder einem pro Gelenk* umschalten.

Hier ist ein Beispiel für die Verwendung zur Anzeige eines Symbols für die Referenzfahrt der Z-Achse:

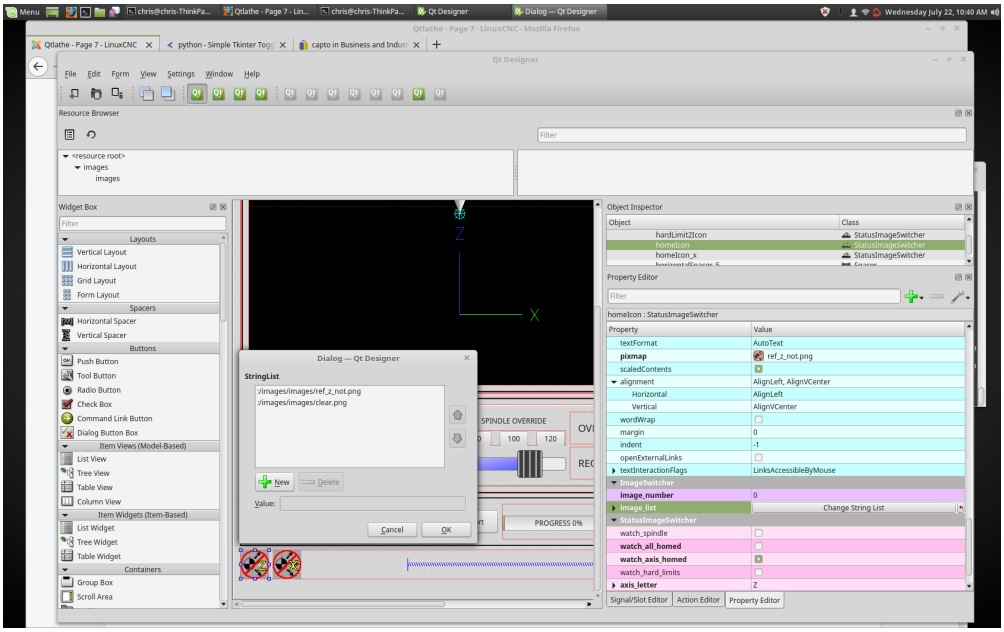


Figure 314. QtVCP `StatusImageSwitcher`: Controller Status Image Switcher

Im Abschnitt zu Eigenschaften ist zu beachten:

- `watch_axis_homed` ist angekreuzt
- `axis_letter` ist auf Z gesetzt

Wenn Sie auf die `image_list` doppelklicken, wird ein Dialog angezeigt, über das Sie Bildpfade hinzufügen können.

Wenn Sie ein Bild als Symbol und ein *klares Bild* haben, sieht es so aus, als ob das Symbol angezeigt und

versteckt wird.

Die Auswahl von Bildpfaden kann durch Auswahl der Eigenschaft `pixmap` und der Auswahl eines Bildes erfolgen.

NOTE	Die Einstellung `pixmap` dient nur zur Testanzeige und wird außerhalb von Qt Designer ignoriert.

- Klicken Sie mit der rechten Maustaste auf den Bildnamen und Sie sollten *Pfad kopieren* sehen.
- Klicken Sie auf *Pfad kopieren* (engl. copy path).
- Doppelklicken Sie nun auf die Eigenschaft *image list* (engl. für Bildliste), damit der Dialog angezeigt wird.
- Klicken Sie auf die Schaltfläche *Neu*.
- Fügen Sie den Bildpfad in das Eingabefeld ein.

Wiederholen Sie dies für das nächste Bild.
Verwenden Sie ein klares Bild, um ein verborgenes Symbol darzustellen.

Sie können die *Anzeige der Bilder* in der Bilderliste *testen*, indem Sie die `image number` (engl. für Bildnummer) ändern. In diesem Fall ist `0` unhomed und `1` würde homed sein.
Dies ist nur für die Testanzeige und wird außerhalb von Qt Designer ignoriert.

`StatusStacked` - Umschaltbares Widget zur Anzeige des Modusstatus

Dieses Widget **zeigt eines von drei Panels an, je nach Modus von LinuxCNC**.

Damit können Sie automatisch verschiedene Widgets in den Modi *Manuell*, *MDI* und *Auto* anzeigen lassen.

TODO

Es basiert auf dem Widget *QStacked* von PyQt.

`ScreenOption` - Widget zum Einstellen allgemeiner Optionen

Dieses Widget fügt dem Bildschirm nichts Visuelles hinzu, sondern **stellt wichtige Optionen ein**.

Dies ist die *bevorzugte Art, diese Optionen zu verwenden*.

Eigenschaften

Diese Eigenschaften können im Qt-Designer, im Python-Handler-Code oder (falls angemessen) in Stylesheets festgelegt werden.

Diese beinhalten:

`halCompBaseName`

Wenn leer, verwendet QtVCP den Namen des Bildschirms als Basisnamen der HAL-Komponente.
Wenn gesetzt, verwendet QtVCP diesen String als Basisnamen der HAL-Komponente.

Wenn die `-c` Kommandozeilenoption beim Laden von QtVCP verwendet wird, dann wird der in der Kommandozeile angegebene Name verwendet - er hat Vorrang vor allen oben genannten Optionen.
Wenn Sie den Basisnamen in der `handlerfile` programmatisch festlegen, werden alle obigen Optionen außer Kraft gesetzt.
Diese Eigenschaft kann nicht in Stylesheets gesetzt werden.

`notify_option`

Einbindung in die Desktop-Benachrichtigungsblasen für Fehler und Meldungen.

`notify_max_messages`

Anzahl der Meldungen, die gleichzeitig auf dem Bildschirm angezeigt werden.

`catch_close_option`

Abfangen des Schließen-Ereignisses, um eine *„Sind Sie sicher“-Eingabeaufforderung* anzuzeigen.

`close_overlay_color`

Farbe der transparenten Ebene, die beim Verlassen angezeigt wird.

`catch_error_option`

Überwachung des LinuxCNC-Fehlerkanals.
Dies sendet auch die Nachricht über `STATUS` an alles, das sich registriert.

`play_sounds_option`

Abspielen von Sounds mit `beep`, `espeak` und dem Systemsound.

`use_pref_file_option`

Einrichten eines *Präferenzdateipfades*.
Die Verwendung des magischen Wortes `WORKINGFOLDER` im Pfad der Einstellungsdatei wird durch den gestarteten Konfigurationspfad ersetzt, z.B. `WORKINFOLDER/my_preferences`.

`use_send_zmq_option`

Wird verwendet, um *ZMQ-basierte ausgehende Nachrichten* zu initiieren.

`use_receive_zmq_messages`

Wird verwendet, um *ZMQ-basierte eingehende Nachrichten* zu initiieren.
Diese Nachrichten *können verwendet werden, um Funktionen in der Handler-Datei aufzurufen*, so dass **externe Programme eng mit QtVCP**-basierten Bildschirmen integriert werden können.

`embedded_program_option`

In der *INI* definierte Programme einbetten.

`default_embed_tab`

Dies ist die Eigenschaft für einen *Standardort zum Einbetten externer Programme*.
Sie sollte auf den Namen eines Registerkarten-Widgets in Qt Designer gesetzt werden.

`focusOverlay_option`

Focus_overlay legt ein transparentes Bild oder ein farbiges Feld über den Hauptbildschirm, um den Fokus auf ein externes Ereignis zu betonen - normalerweise ein Dialog.

messageDialog_option

Richtet den Nachrichtendialog ein - wird für allgemeine Nachrichten verwendet.

message_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der Nachrichtendialog eingeblendet wird.

closeDialog_option

Richtet den Standarddialog zum Schließen des Bildschirms ein.

entryDialog_option

Richtet den numerischen Eingabedialog ein.

entryDialogSoftKey_option

Richtet eine schwebende Softwaretastatur ein, wenn der Eingabedialog fokussiert ist.

entry_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der Eingabedialog angezeigt wird.

toolDialog_option

Richtet den manuellen Werkzeugwechsel-Dialog ein, inklusive HAL-Pin.

tool_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der Werkzeugdialog angezeigt wird.

ToolUseDesktopNotify

Option zur Verwendung von Desktop-Benachrichtigungsdialogen für manuelle Werkzeugwechseldialoge.

ToolFrameless

Framesless-Dialoge können von Benutzern nicht einfach verschoben werden.

fileDialog_option

Richtet den Dateiauswahldialog ein.

file_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der Dateidialog angezeigt wird.

keyboardDialog_option

Richtet ein Tastatureingabe-Widget ein.

keyboard_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der Tastaturdialog angezeigt wird.

vesaProbe_option

Richtet den Versa-Style-Probe-Dialog ein.

versaProbe_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der Dialog versaProbe angezeigt wird.

macroTabDialog_option

legt den Makro-Auswahldialog fest.

macroTab_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der macroTab-Dialog angezeigt wird.

camViewDialog_option

Richtet den Kameraausrichtungsdialog ein.

camView_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der Dialog camView angezeigt wird.

toolOffset_option

Richtet das Dialogfeld für die Anzeige/Editierung von Werkzeugkorrekturen ein.

toolOffset_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der toolOffset-Dialog angezeigt wird.

originOffset_option

Richtet das Dialogfeld für die Anzeige/Editierung des Ursprungs ein.

originOffset_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der originOffset-Dialog angezeigt wird.

calculatorDialog_option

Richtet das Eingabefenster für den Taschenrechner ein.

calculator_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn das Dialogfeld "Rechner" angezeigt wird.

machineLogDialog_option

Richtet einen Dialog ein, um Protokolle von der Maschine und QtVCP anzuzeigen.

machineLog_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der machineLog-Dialog angezeigt wird.

runFromLineDialog_option

Richtet einen Dialog ein, der die Startoptionen anzeigt, wenn die Maschinenausführung von einer beliebigen Zeile aus gestartet wird.

runFromLine_overlay_color

Farbe der transparenten Ebene, die angezeigt wird, wenn der runFromLine-Dialog angezeigt wird.

user1Color

Optionale Farbe die Screen Designer in nutzen können.

user2Color

Optionale Farbe die Screen Designer in nutzen können.

user3Color

Optionale Farbe die Screen Designer in nutzen können.

user4Color

Optionale Farbe die Screen Designer in nutzen können.

user5Color

Optionale Farbe die Screen Designer in nutzen können.

user6Color

Optionale Farbe die Screen Designer in nutzen können.

user7Color

Optionale Farbe die Screen Designer in nutzen können.

user8Color

Optionale Farbe die Screen Designer in nutzen können.

user9Color

Optionale Farbe die Screen Designer in nutzen können.

user10Color

Optionale Farbe die Screen Designer in nutzen können.

Eigenschaften programmgesteuert festlegen

Der Screendesigner wählt die **Standardeinstellungen des Widgets `screenOptions` aus**.

Einmal ausgewählt, müssen die meisten nicht mehr geändert werden. Bei Bedarf können jedoch einige in der Handler-Datei oder in Stylesheets geändert werden.

- **In der Handler-Datei**:
  Hier referenzieren wir das Widget durch den benutzerdefinierten Namen des Qt-Designers:

```
# red,green,blue,alpha 0-255
color = QtGui.QColor(0, 255, 0, 191)
self.w.screen_options.setProperty('close_overlay_color', color)
self.w.screen_options.setProperty('play_sounds_option',False)
```

- **In Stylesheets**:
  Hier können wir das Widget durch den benutzerdefinierten Namen von Qt Designer referenzieren oder nach Widget-Klassenname.

```
/* red, green, blue 0-255, alpha 0-100% or 0.0 to 1.0 */
/* the # sign is used to refer to Qt Designer defined widget name */
/* matches/applied to only this named widget */
#screen_options {
  qproperty-close_overlay_color: rgba(0, 255, 0, 0.75)
}
```

Einige Einstellungen werden nur beim Start geprüft und verursachen daher keine Änderungen nach dem Start. In diesen Fällen müssen Sie die Änderungen *nur* im Qt Designer vornehmen.

Einträge in der Einstellungsdatei

Wenn die Option *Voreinstellungsdatei* ausgewählt ist, erstellt das Widget `screenOption` eine **INI-basierte Voreinstellungsdatei**.

Während *andere QtVCP Widgets diese Liste ergänzen*, fügt das `screenOptions` Widget diese Einträge unter den folgenden Überschriften hinzu:

[SCREEN_OPTIONS]

catch_errors ***(bool)***

desktop_notify ***(bool)***

Ob Fehler/Meldungen im Benachrichtigungsmechanismus des Systems angezeigt werden sollen.

notify_max_msgs ***(int)***

Anzahl der angezeigten Fehler zu einem Zeitpunkt.

shutdown_check ***(bool)***

Ob ein Bestätigungsdialog erscheinen soll.

sound_player_on ***(bool)***

Schaltet alle Töne ein oder aus.

[MCH_MSG_OPTIONS]

mchnMsg_play_sound ***(bool)***

Um Warntöne wiederzugeben, wenn ein Dialogfeld angezeigt wird.

mchnMsg_speak_errors ***(bool)***

Verwendung von Espeak, um Fehlerbotschaften zu sprechen.

mchnMsg_speak_text ***(bool)***

Espeak verwenden, um alle anderen Nachrichten zu sprechen.

mchnMsg_sound_type ***(str)***

Ton, der abgespielt wird, wenn Nachrichten angezeigt werden. Siehe Hinweise unten.

[USER_MSG_OPTIONS]

usermsg_play_sound ***(bool)***

Um Warntöne wiederzugeben, wenn ein Dialogfeld angezeigt wird.

userMsg_sound_type ***(str)***

Ton, der abgespielt wird, wenn Benutzermeldungen angezeigt werden. Siehe Hinweise unten.

userMsg_use_focusOverlay ***(bool)***

[SHUTDOWN_OPTIONS]

shutdown_play_sound ***(bool)***

shutdown_alert_sound_type ***(str)***

Ton, der abgespielt wird, wenn Nachrichten angezeigt werden. Siehe Hinweise unten.

shutdown_exit_sound_type ***(str)***

Ton, der abgespielt wird, wenn Nachrichten angezeigt werden. Siehe Hinweise unten.

shutdown_msg_title ***(str)***

Kurzer Titelstring, der im Dialog angezeigt wird.

shutdown_msg_focus_text ***(str)***

Großer Text, der in die Fokusebene eingeblendet wird.

shutdown_msg_detail ***(str)***

Längere beschreibende Zeichenfolge zur Anzeige im Dialog.

NOTIFY_OPTIONS

notify_start_greeting ***(bool)***

Ob ein Begrüßungsdialog beim Start angezeigt werden soll.

notify_start_title ***(str)***

Kurzer Titelstring.
Wenn die Option "Sprechen" ebenfalls ausgewählt ist, wird der Titel mit Espeak gesprochen.

notify_start_detail ***(str)***

Längere Zeichenfolge zur Beschreibung.

notify_start_timeout ***(int)***

Zeit in Sekunden bis zur Anzeige vor dem Schließen.

`*_sound_type` *Einträge*

- **Systemklänge**.
  In Debian/Ubuntu/Mint-basierten Installationen sollten diese *Systemklänge* als Sound-Typ-Einträge oben verfügbar sein:
  - `ERROR`
  - `READY`
  - `DONE`
  - `ATTENTION`
  - `RING`

- `LOGIN`
- `LOGOUT`
- `BELL` (engl. für Glocke)

Diese Sound-Optionen erfordern die Installation von `python3-gst1.0`.

- **Audiodateien**
  Sie können auch einen *Dateipfad für eine beliebige Audiodatei* angeben.
  Sie können `~` im Pfad verwenden, um den Pfad der Benutzer-Home-Datei zu ersetzen.
- **Kernel-Pieptöne**
  Wenn das *Kernel-Modul* `beep` installiert und nicht deaktiviert ist, sind diese Sound-Einträge zur Verfügung:
  - `BEEP`
  - `BEEP_RING`
  - `BEEP_START`
- **Text-To-Speech**
  Wenn das *Espeak* Modul (`python3-espeak`) installiert ist, können Sie den Eintrag `SPEAK` verwenden, um Text vorlesen zu lassen:
- **`SPEAK '_meine Nachricht_'`**

`StatusSlider` - Controller-Einstellungs-Schieberegler-Widget

Mit diesem Widget kann der Benutzer **eine LinuxCNC-Einstellung mit einem Schieberegler** anpassen.

Die Kurzbeschreibung kann folgendes anpassen:

- Jogging-Rate
- Winkel-Jog-Rate
- Vorschubgeschwindigkeit
- Spindel-Override-Rate
- Eilgang Übersteuerungsrate (engl. rapid override rate)

Eigenschaften

`StatusSlider` hat die folgenden Eigenschaften:

`halpin_option`

Sets option to make a HAL real pin that reflects current value.

`rapid_rate`

Wählt einen Schieberegler für die Übersteuerungsrate des Eilgangs.

`feed_rate`

Wählt einen Schieberegler für die Vorschub-Neufestsetzung (engl. feed override rate).

spindle_rate

Wählt einen Schieberegler für die Übersteuerungsrate des Eilgangs.

jograte_rate

Wählt einen linearen Jograte-Schieberegler aus.

jograte_angular_rate

Wählt einen eckigen Jograte-Schieberegler aus.

max_velocity_rate

Wählt einen Schieberegler für die maximale Geschwindigkeitsrate aus.

alertState

String zum Definieren der Stiländerung: `read-only` (enlg. nur lesen), `under` (engl. für unter), `over` (engl. für über) und `normal`.

alertUnder

Legt den Float-Wert fest, der dem Stylesheet eine "Unter"-Warnung signalisiert.

alertOver

Legt den Gleitkommawert fest, der dem Stylesheet die Warnung "Über" signalisiert.

Diese können eingestellt werden in:

- Qt Designer
- Python-Handler-Code,

```
self.w.status_slider.setProperty('spindle_rate',True)
self.w.status_slider.setProperty('alertUnder',35)
self.w.status_slider.setProperty('alertOver',100)
```

- Oder (gegebenenfalls) in Stylesheets.

```
/* Warnfarben für Übersteuerungen, wenn sie außerhalb des normalen Bereichs liegen*/
/* Name des Widget-Objekts ist slider_spindle_ovr */

#slider_spindle_ovr[alertState='over'] {
    background: red;
}
#slider_spindle_ovr[alertState='under'] {
    background: yellow;
}
```

Es basiert auf PyQt's *QSlider*.

`StateLED` - Controller-Status-LED-Widget

Dieses Widget zeigt den **Status des ausgewählten LinuxCNC-Status** an.

Zustände

Die Statusoptionen sind:

is_paused_status

is_estopped_status

is_on_status

is_idle_status_

is_homed_status

is_flood_status

is_mist_status

is_block_delete_status

is_optional_stop_status

is_joint_homed_status

is_limits_overridden_status

is_manual_status

is_mdi_status

is_auto_status

is_spindle_stopped_status

is_spindle_fwd_status

is_spindle_rev_status

is_spindle_at_speed_status

is_neg_limit_tripped

is_pos_limit_tripped

is_limits_tripped

Eigenschaften

Es gibt Eigenschaften, die geändert werden können:

halpin_option

Fügt einen Ausgangspin hinzu, der den ausgewählten Zustand wiedergibt.

invert_state_status

Invertieren des LED-Status im Vergleich zum LinuxCNC-Status.

diameter

Durchmesser der LED.

color

Farbe der LED im eingeschalteten Zustand.

off_color

Farbe der LED im ausgeschalteten Zustand.

`alignment`

Qt-Hinweis zur Ausrichtung.

`state`

Aktueller Zustand der LED (zum Testen in Qt Designer).

`flashing`

Schaltet die Blinkoption ein und aus.

`flashRate`

Legt die Blitzrate fest.

Die LED-Eigenschaften können in einem Stylesheet mit folgendem Code definiert werden, welcher der Datei `.qss` hinzugefügt wird.

```
State_LED #name_of_led{       ①
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
}
```

① `name_of_led` wäre der im Editor von Qt Designer definierte Name.

Es basiert auf dem *LED*-Widget.

`StatusAdjustmentBar` - Widget zum Einstellen von Controller-Werten

Dieses Widget ermöglicht die **Einstellung von Werten über Schaltflächen, während ein Balken angezeigt wird**.

Außerdem gibt es einen *optionalen Hoch/Tief-Knopf*, der gedrückt gehalten werden kann, um die **Stufen** einzustellen.

Die Kurzbeschreibung kann folgendes anpassen:

- Jogging-Rate
- Winkel-Jog-Rate
- Vorschubgeschwindigkeit
- Spindel-Override-Rate
- Eilgang Übersteuerungsrate (engl. rapid override rate)

Sie basiert auf PyQt's *QProgressBar*.

`SystemToolButton` - Widget zur Auswahl des Benutzersystems

Mit diesem Widget können Sie **manuell ein G5x Benutzersystem auswählen, indem Sie es gedrückt halten**.

Wenn Sie den Text der Schaltfläche nicht festlegen, wird sie automatisch auf das aktuelle System aktualisiert.

Es basiert auf PyQts *QToolButton*.

`StateEnableGridlayout` - Controller State Enabled Container Widget

```
_Deaktivieren Sie die Widgets darin je nach LinuxCNCs aktuellem Zustand_.
```

Dies ist ein **Container, in dem andere Widgets platziert werden können**.

Eingebettete Widgets werden ausgegraut, wenn die Funktion `StateEnableGridlayout` deaktiviert ist.

Es kann selektiv reagieren auf:

- Maschine ein
- Interpreter idle
- Notaus aus
- Alle referenziert (engl. all-homed)

Es basiert auf PyQt's *QGridLayout*.

`StatusImageSwitcher` - Controller-Statusbild-Umschalt-Widget

Dieses Widget wird **Bilder basierend auf dem LinuxCNC-Status anzeigen**.

Sie können sehen:

- Den Zustand der Spindel,
- den Status der Referenzierung aller Achsen,
- den Zustand der Referenzierung einer bestimmten Achse,
- den Zustand der harten Grenzen.

Es basiert auf PyQt's FIXME

`ToolOffsetView` - Widget zum Anzeigen und Bearbeiten von Werkzeug-Offsets

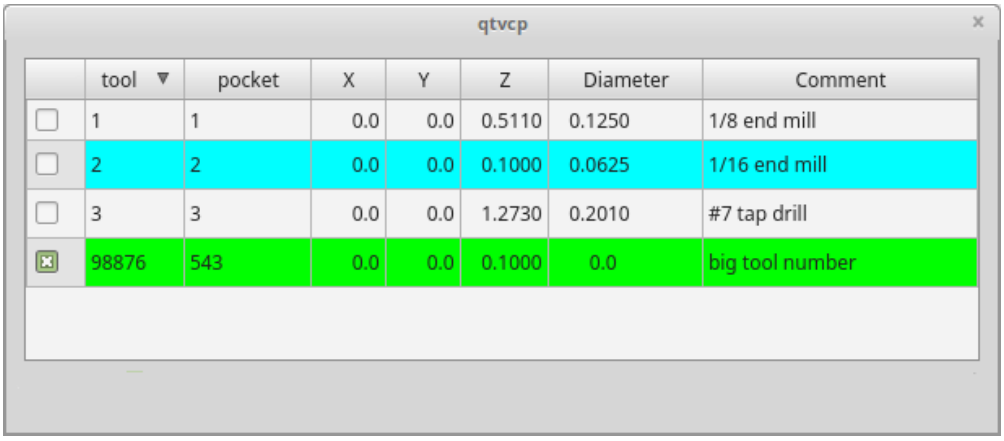


Figure 315. QtVCP `ToolOffsetView` - Widget zum Anzeigen und Bearbeiten von Werkzeug-Offsets

Dieses Widget **zeigt Werkzeug-Offsets an und ermöglicht deren Änderung**.

Es wird *die Werkzeugtabelle von LinuxCNC* für vorgenommene oder gefundene Änderungen aktualisieren.

Die Einstellungen können in LinuxCNC nur nach der Referenzfahrt und im Ruhezustand des Motion Controllers geändert werden.

Die Anzeige und der Eintrag wechseln zwischen metrisch und imperial, basierend auf der *aktuellen G20 /G21* Einstellung von LinuxCNC.

Das aktuell verwendete Werkzeug wird hervorgehoben, und das aktuell ausgewählte Werkzeug wird in einer anderen Farbe hervorgehoben.

Das Kontrollkästchen neben jedem Werkzeug kann auch für eine *Aktion* ausgewählt werden, die von zusätzlichem Code abhängt, der entweder zu einem kombinierten Widget, wie dem Dialogfeld `ToolOffsetView` oder dem Code des Bildschirmhandlers, hinzugefügt wird.
Typische Aktionen sind "Ausgewähltes Werkzeug laden", "Ausgewählte Werkzeuge löschen", usw.

Wenn Sie auf die Spalten und Zeilen klicken, können Sie die Einstellungen anpassen.

Für die Daten- oder Texteingabe kann ein Dialogfeld eingeblendet werden.

Der Kommentarbereich (engl. comments section) wird in der Regel im Dialog für den manuellen Werkzeugwechsel angezeigt.

Bei Verwendung einer *Drehbankkonfiguration* kann es Spalten für X- und Z-Verschleiß geben.

Um diese Spalten zur Einstellung des *Werkzeugverschleißes* zu verwenden, ist eine neu zugeordnete Werkzeugwechselroutine erforderlich.

Es basiert auf PyQt's *QTableView*, *QAbstractTableModel*, und *ItemEditorFactory*.
Eigenschaften, Funktionen und Stile der PyQt-Basisobjekte sind immer verfügbar.

Eigenschaften

`ToolOffsetView` hat Eigenschaften, die im Qt Designer, im Python-Handler-Code oder (falls zutreffend) in Stylesheets eingestellt werden können:

dialog_code_string

Legt fest, welcher Dialog bei einer numerischen Eingabe erscheint.

text_dialog_code_string

Legt fest, welcher Dialog bei der Texteingabe angezeigt wird.

metric_template

Metrisches numerisches Datenformat.

imperial_template

Imperiales numerisches Datenformat.

styleCodeHighlight

Hervorhebungsfarbe des aktuellen Werkzeugs.

styleCodeSelected

Ausgewählte Hervorhebungsfarbe.

In einer Handler-Datei:

```
self.w.tooloffsetview.setProperty('dialog_code','CALCULATOR')
self.w.tooloffsetview.setProperty('metric_template','%10.3f')
```

und in Stylesheets:

```
ToolOffsetView{
  qproperty-styleColorHighlist: lightblue;
  qproperty-styleColorSelected: #444;
}
```

Funktionen

Die Funktion `ToolOffsetView` hat einige Funktionen, die für Screenbuilder nützlich sind, um Aktionen hinzuzufügen:

add_tool()

Fügt ein leeres Dummy-Werkzeug (99) hinzu, das der Benutzer nach Belieben bearbeiten kann.

delete_tools()

Löscht die in der Checkbox ausgewählten Werkzeuge.

get_checked_list()

Gibt eine Liste der durch Ankreuzfelder ausgewählten Werkzeuge zurück.

set_all_unchecked()

Hebt die Markierung aller ausgewählten Werkzeuge auf.

Beispiel für eine Handler-Datei, welche die oben genannten Funktionen ausführt.

```
self.w.tooloffsetview.add_tool()
self.w.tooloffsetview.delete_tools()
toolList = self.w.tooloffsetview.get_checked_list()
self.w.tooloffsetview.set_all_unchecked()
```

`VersaProbe` - Fräsen-Tast-Widget

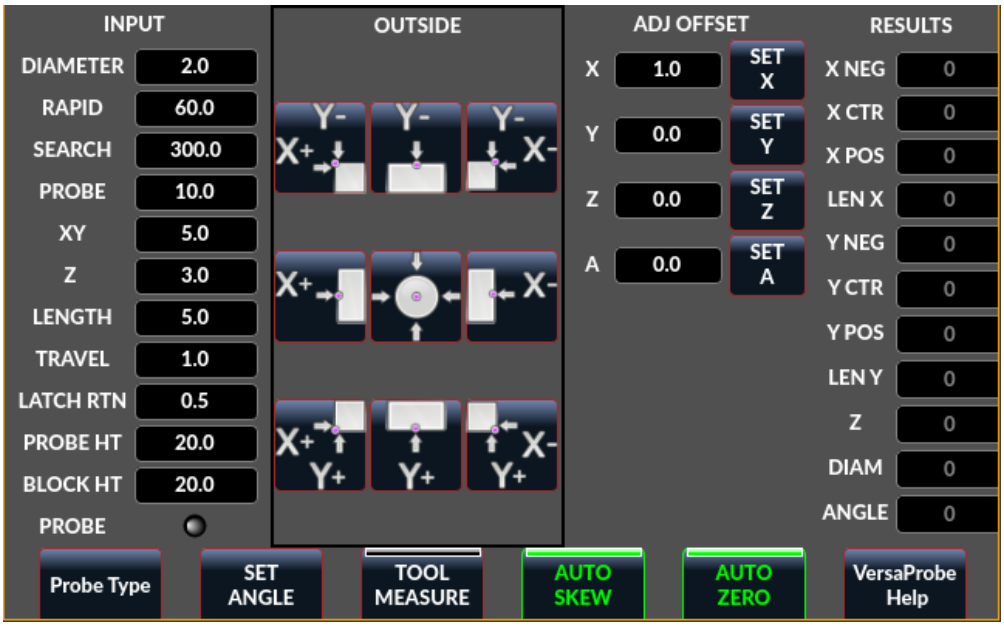


Figure 316. QtVCP `VersaProbe`: Fräsen-Tast-Widget

Widget zum **Sondieren auf einer Fräse**. Wird vom *QtDragon*-Bildschirm verwendet.

12.8.3. Dialog-Widgets

Dialoge werden verwendet, um **unmittelbar benötigte Informationen** gezielt darzustellen oder abzufragen.

Die typischerweise verwendeten Dialoge können mit dem *ScreenOptions widget* geladen werden.

Sie können sie auch direkt zur *UI* hinzufügen - allerdings muss jedes Dialogfeld einen eindeutigen Startnamen haben, sonst werden mehrere Dialogfelder nacheinander angezeigt.

Dialoge aus Python-Code verwenden

Sie können Dialoge direkt mit *Python-Code* anzeigen, aber eine sicherere Methode ist die **Verwendung von `STATUS`-Nachrichten**, um den Dialog zu starten und die gesammelten Informationen zurückzugeben.

- **Registrierung auf dem Kanal `STATUS`**:
  Um dies einzurichten, registrieren Sie sich zuerst, um die `allgemeine` Nachricht von `STATUS` ZU EMPFANGEN:

  ```
  STATUS.connect('general',self.return_value)
  ```

- **Fügen Sie eine Funktion hinzu, um einen Dialog aufzurufen**:
  Diese Funktion muss *einen Nachrichten-`dict` anlegen, das an die dialog übergeben* wird.
  Diese Nachricht wird in der allgemeinen Nachricht mit dem Zusatz einer *Rückgabe-Variablen* zurückgegeben.
  Es ist möglich, der Nachricht *extra Benutzer information* hinzuzufügen. Der Dialog ignoriert diese und gibt sie zurück.

  `NAME`

  Startet den Codenamen des anzuzeigenden Dialogs.

  `ID`

  Eine eindeutige ID, damit wir nur den angeforderten Dialog bearbeiten.

  `TITLE`

  Der Titel, der für das Dialogfeld verwendet werden soll.

```
def show_dialog(self):
    mess = {'NAME':'ENTRY','ID':'__test1__',
            'TITLE':'Test Entry'}
    ACTION.CALL_DIALOG, mess)
```

- **Hinzufügen einer Callback-Funktion, um die allgemeine Nachricht zu verarbeiten**:
  Beachten Sie, dass diese Funktion *alle allgemeinen Nachrichten* abruft, so dass die `dict`-Keynamen sind nicht garantiert vorhanden sind. Es ist ratsam, die Funktion `.get()` zu benutzen und/oder `try` /`except` zu verwenden. Diese Funktion sollte:

  - Prüfen Sie, ob der Name und die Kennung mit den Angaben übereinstimmen, die wir gesendet haben,
  - Extrahieren Sie dann den Rückgabewert und alle Benutzervariablen.

```
# Verarbeitung der STATUS return message
def return_value(self, w, message):
    rtn = message.get('RETURN')
    code = bool(message.get('ID') == '__test1__')
    name = bool(message.get('NAME') == 'ENTRY')
    if code and name and not rtn is None:
        print('Entry return value from {} = {}'.format(code, rtn))
```

`LcncDialog` - Allgemeines Nachrichtendialog-Widget

Dies ist ein **Allgemeines Nachrichten-Dialog-Widget**.

Wenn ein Fokus-Overlay-Widget vorhanden ist, kann es signalisieren, dass es angezeigt werden soll.

Wenn die Klangbibliothek eingerichtet ist, kann sie Klänge *abspielen*.

Es gibt *Optionen*, die gesetzt werden können, wenn ein Dialog angefordert wird, diese würden der Nachricht `dict` hinzugefügt.

TITLE

Titel des Dialogfensters.

MESSAGE

Titel Nachrichtentext in Fettdruck.

MORE

Standardtext unter der Überschrift.

DETAILS

Ursprünglicher versteckter Text.

TYPE ***(OK|YESNO|OKCANCEL)***

ICON **(QUESTION|INFO|CRITICAL|WARNING)**

PINNAME

Noch nicht implementiert.

FOCUSTEXT ***(overlay text|None)***

Text, der angezeigt werden soll, wenn das Fokus-Overlay verwendet wird. Verwenden Sie `None` für keinen Text.

FOCUSCOLOR **(QColor(_R, G, B, A_))**

Farbe, die verwendet werden soll, wenn das Fokus-Overlay verwendet wird.

PLAYALERT

Abzuspielender Ton, falls vorhanden, z.B. `SPEAK` <*Spoken_message*> .

Bei der Verwendung der Funktion "Abfrage-Dialog" von "STATUS" ist der "Standard-Startname" **MESSAGE**.

Sie basiert auf der *QMessagebox* von PyQt.

`ToolDialog` - Dialog-Widget für den manuellen Werkzeugwechsel

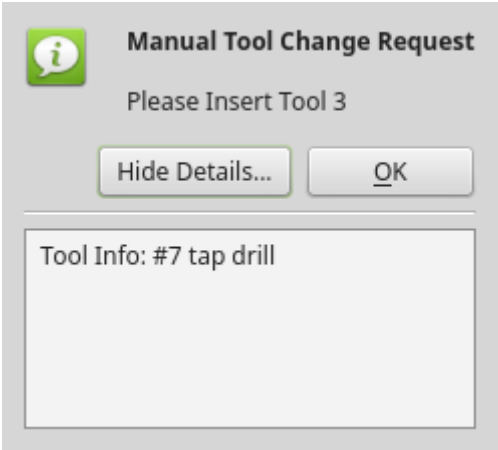


Figure 317. QtVCP `ToolDialog`: Dialog zum manuellen Werkzeugwechsel

Dies wird als Aufforderung zum **manuellen Werkzeugwechsel** verwendet.

Es verfügt über *HAL Pins, die mit dem controller* der Maschine verbunden werden können. Die Pins haben den gleichen Namen wie die ursprüngliche AXIS manuelle Werkzeugeingabeaufforderung und funktionieren gleich.

Der Werkzeugwechsel-Dialog kann *nur über HAL-Pins* aufgerufen werden.

Wenn ein Fokus-Overlay-Widget vorhanden ist, signalisiert es, dass es angezeigt werden soll.

Sie basiert auf der *QMessagebox* von PyQt.

`FileDialog` - Dialog Widget zum Laden und Speichern von Dateien

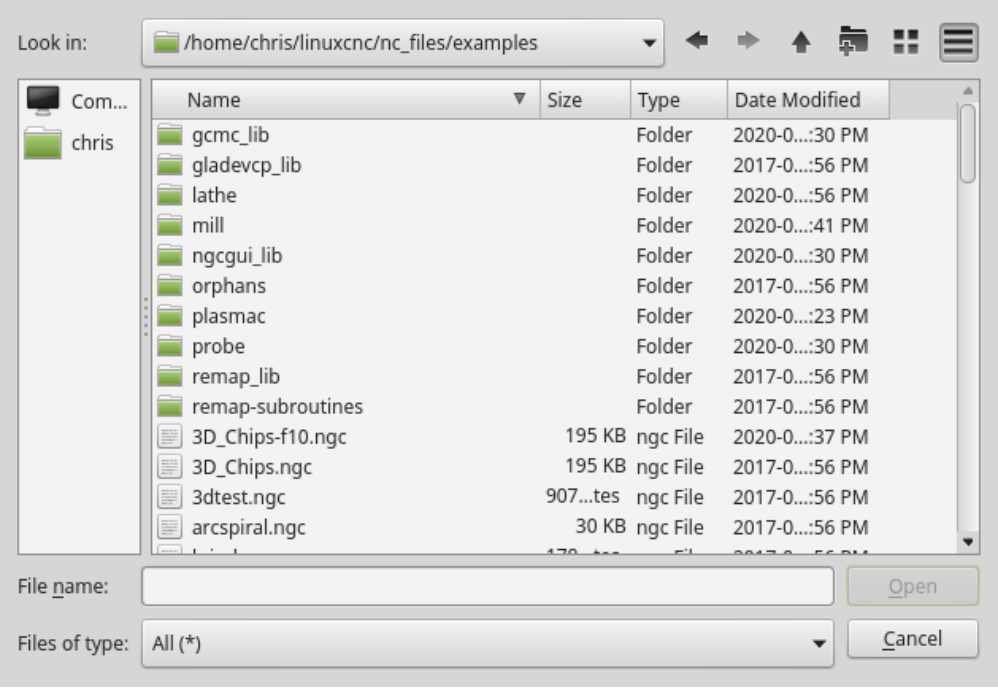


Figure 318. QtVCP `FileDialog`: Widget für das Laden und Speichern von Dateien

Dies wird zum **Laden von G-Code-Dateien** verwendet.

Wenn ein Fokus-Overlay-Widget vorhanden ist, signalisiert es, dass es angezeigt werden soll.

Wenn Sie die `Request-Dialog`-Funktion von `STATUS` verwenden, lauten die Standard-Startnamen **`LOAD`** oder **`SAVE`**.

Es gibt *options*, die beim Anfordern eines Dialogs gesetzt werden können, diese würden dem Meldungsdiktat hinzugefügt:

`EXTENSIONS`

`FILENAME`

`DIRECTORY`

Ein Beispiel für einen Python-Aufruf, für einen *Load-Dialog*:

```
mess = {'NAME':'LOAD','ID':'_MY_DIALOG_',
            'TITLE':'Load Some text File',
```

```
            'FILENAME':'~/linuxcnc/nc_files/someprogram.txt',
            'EXTENSIONS':'Text Files (*.txt);;ALL Files (*.*)'
            }
ACTION.CALL_DIALOG(mess)
```

Und für einen *Speicherdialog*

```
mess = {'NAME':'SAVE','ID':'_MY_DIALOG_',
            'TITLE':'Save Some text File',
            'FILENAME':'~/linuxcnc/nc_files/someprogram.txt',
            'EXTENSIONS':'Text Files (*.txt);;ALL Files (*.*)'
            }
ACTION.CALL_DIALOG(mess)
```

Sie basiert auf der *QMessagebox* von PyQt.

`OriginOffsetDialog` - Dialogfeld-Widget für die Einstellung des Ursprungsversatzes

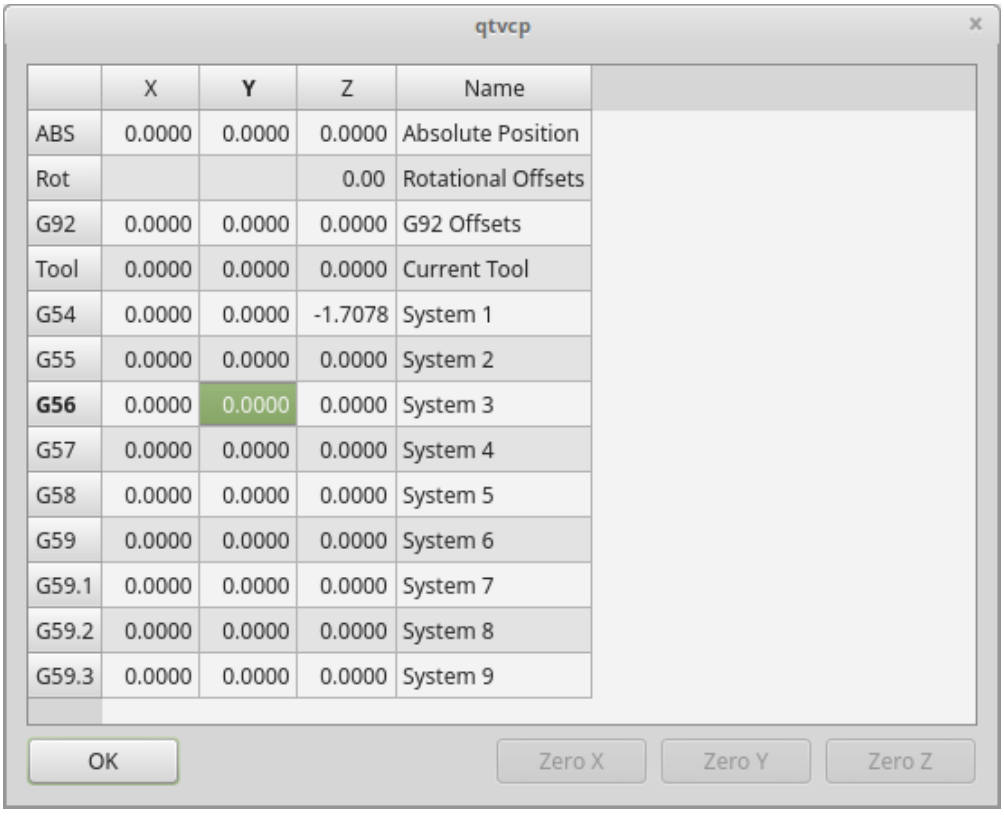


Figure 319. QtVCP `OriginOffsetDialog`: Widget zur Einstellung des Ursprungsversatzes

Mit diesem Widget kann man **die Nullpunktverschiebung des Benutzersystems direkt** in einem Dialogformular ändern.

Wenn ein Fokus-Overlay-Widget vorhanden ist, wird es angezeigt.

Bei Verwendung der `request-dialog`-Funktion von `STATUS` ist der standardmäßige Startname **`ORIGINOFFSET`**.

Es basiert auf PyQts *QDialog*.

`ToolOffsetDialog` - Dialogfenster-Widget zur Einstellung des Werkzeugversatzes

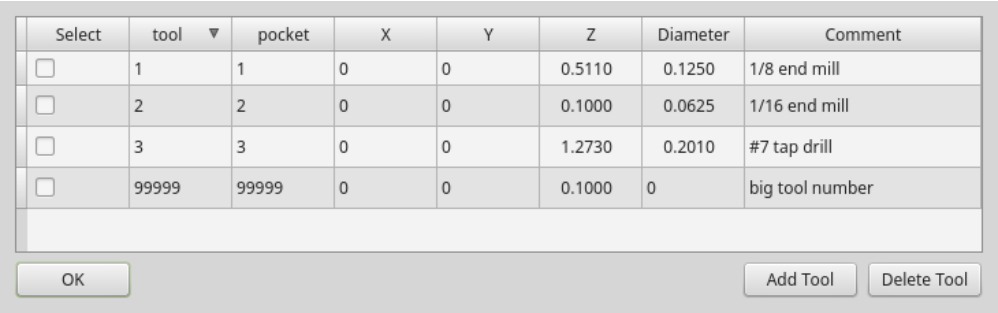


Figure 320. QtVCP `ToolOffsetDialog`: Werkzeug-Offset-Einstellungsdialog-Widget

Mit diesem Widget kann man die **Werkzeugversätze direkt** in einem Dialogformular **ändern**.

Wenn ein Fokus-Overlay-Widget vorhanden ist, wird es angezeigt.

Bei Verwendung der `request-dialog`-Funktion von `STATUS` ist der standardmäßige Startname **`TOOLOFFSET`**.

Es basiert auf PyQts *QDialog*.

`ToolChooserDialog` - Werkzeug-Auswahl Dialog Widget

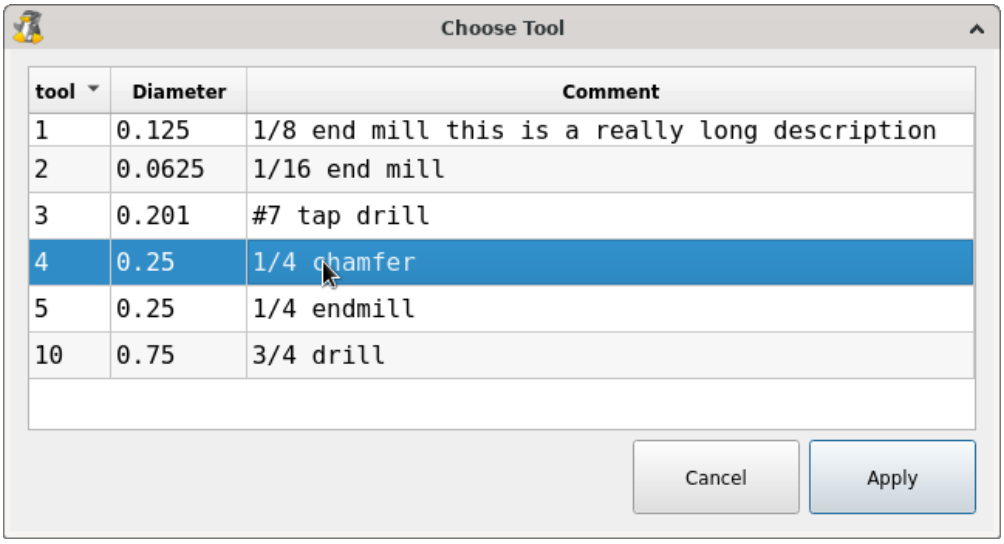


Figure 321. QtVCP `ToolChooserDialog`: Werkzeug-Auswahl Dialog Widget

Dieses Widget ermöglicht es dem Bediener, eines der in der Werkzeugtabelle definierten Werkzeuge auszuwählen. Wird ein Werkzeug ausgewählt und „Übernehmen“ gedrückt oder das Werkzeug doppelt angeklickt, gibt der Dialog die ausgewählte Werkzeugnummer zurück. Dies kann beispielsweise in Verbindung mit dem Widget `OperatorValueLine` verwendet werden, um ein Werkzeugwechsel-Widget zu erstellen.

Wenn ein Fokus-Overlay-Widget vorhanden ist, wird es angezeigt.

Bei Verwendung der `request-dialog`-Funktion von `STATUS` ist der voreingestellte Startname **`TOOLCHOOSER`**.

Es basiert auf PyQts *QDialog*.

MachineLog - Widget für die Anzeige des Journals für Maschinenereignisse

Dieses Widget zeigt verschiedene **Protokoll-Ereignismeldungen** an, die vom System während der aktuellen Sitzung ausgegeben wurden. Dazu gehören sowohl Informationsmeldungen als auch Fehler.

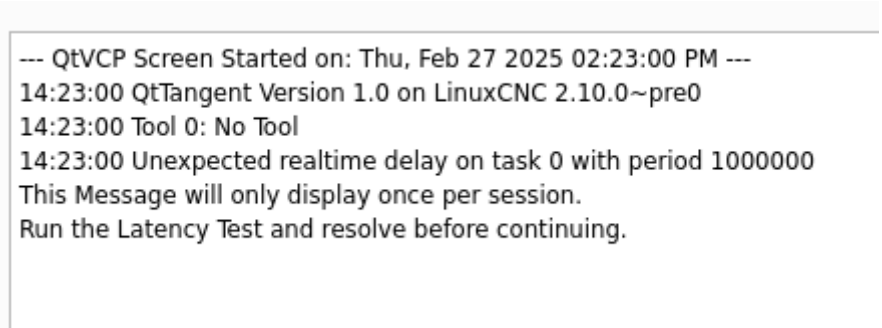


Figure 322. QtVCP `MachineLog`: Maschinen-Ereignisse (engl. events) Log in `machine_log` (plain) Modus

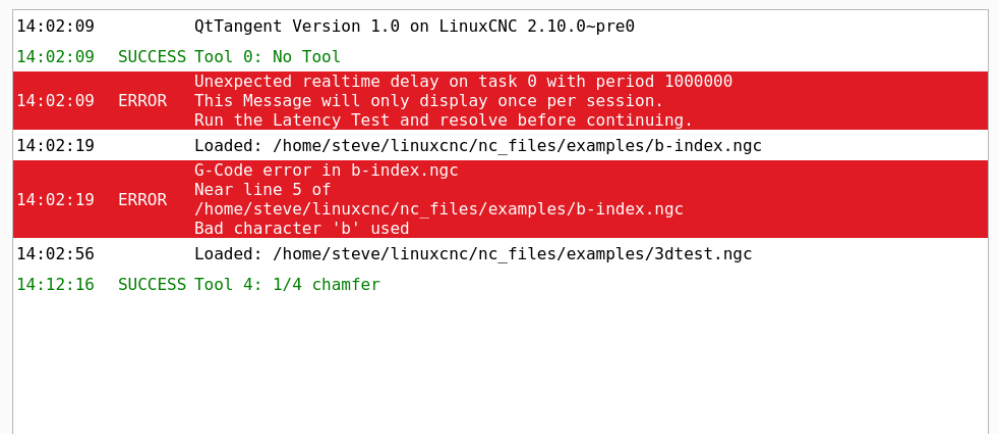


Figure 323. QtVCP `MachineLog`: Maschinen-Events Log im `machine_log_severity` Modus

Es können zwei verschiedene Arten von Protokollen angezeigt werden:

- maschine log (einfache Text-Darstellung, oder Scheweregrad hervorgehoben)
- integrator log (nur einfache Text-Darstellung)

Der vom Widget angezeigte Logtyp wird durch die Optionseigenschaften des Widgets gesteuert. Durch die Auswahl von `machine_log_option` oder `integrator_log_option` wird das entsprechende Log angezeigt. Diese Optionen werden in einem Qt `QTextEdit` Widget mit einfachen Stilprotokollen angezeigt.

Zusätzlich gibt es eine Eigenschaft `machine_log_severity_option`, die gewählt werden kann, über die das Maschinenprotokoll in einer Vielzahl von Farben je nach Schwere der Nachricht anzeigen wird, indem ein `QTableWidget` verwendet wird. Die Farben können mit den Eigenschaften des Widgets konfiguriert werden.

Abgeben von Log Meldungen mit Schweregrad

Der Schweregrad wird über den `option`-Wert vermittelt, der zusammen mit dem `STATUS`-Signal `update-machine-log` gesendet wird. Der Parameter `option` ist eine durch Kommata getrennte Liste und enthält typischerweise

```
text = 'an error has occurred.'
STATUS.emit('update-machine-log', text, 'TIME,ERROR')
```

Löschen des Logs

Das Protokoll kann durch Aufrufen der `clear()`-Methode des Widgets gelöscht werden.

`MacroTabDialog` - Dialog-Widget zum Starten von Makros

Dies ist ein Dialog zum **Anzeigen des Makrotab-Widgets**.

`MacroTab` zeigt eine *Auswahl von Makroprogrammen an, die mit Symbolen ausgeführt werden.*

Wenn ein Fokus-Overlay-Widget vorhanden ist, signalisiert es, dass es angezeigt werden soll.

Bei Verwendung der ``STATUS`` `request-dialog`-Funktion lautet der Standardstartname **`MACROTAB`**.

Es basiert auf PyQts *QDialog*.

`CamViewDialog` - Dialogfeld-Widget für die WebCam-Werkstück-Ausrichtung

Dies ist ein Dialog zur **Anzeige des CamView-Widgets für die Ausrichtung von Werkstücken mit der Webcam**.

Bei Verwendung der ``STATUS`` `request-dialog`-Funktion lautet der Standardstartname **`CAMVIEW `**.

Es basiert auf PyQts *QDialog*.

`EntryDialog` - Widget zum Bearbeiten des Zeilendialogs

Dies ist ein Dialog zum **Anzeigen einer Bearbeitungszeile für die Informationseingabe**, wie z. B. den Versatz zum Ursprung (engl. origin offset).

Es liefert den Eintrag über `STATUS`-Nachrichten unter Verwendung eines Python `DICT`.

Das `DICT` enthält mindestens den Namen des angeforderten Dialogs und einen ID-Code.

Bei Verwendung der ``STATUS`` `request-dialog`-Funktion lautet der Standardstartname **`ENTRY `**.

Es basiert auf PyQts *QDialog*.

`CalculatorDialog` - Rechner-Dialog-Widget

0Dies ist ein Dialog, um einen **Taschenrechner für die numerische Eingabe** anzuzeigen, z. B. für Nullpunktverschiebung, Spindeldrehzahl etc. Er ist hauptsächlich für die Nutzung per Touchscreen vorgesehen, unterstützt aber auch die Eingabe über eine physische Tastatur.

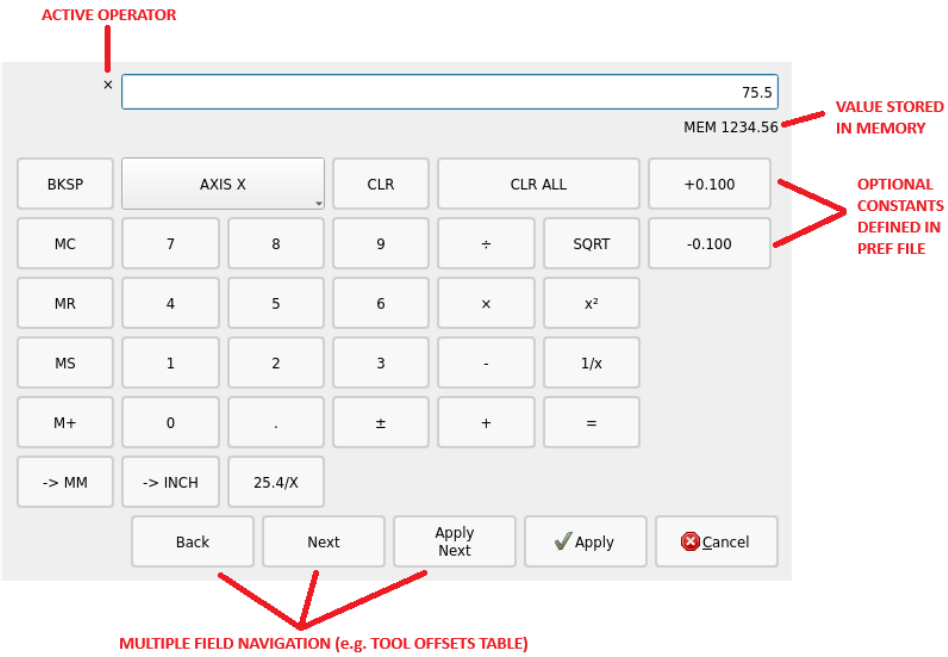


Figure 324. QtVCP `CalculatorDialog`: Rechner-Dialog-Widget

Es liefert den Eintrag über `STATUS`-Nachrichten unter Verwendung eines Python `DICT`.

Das `DICT` enthält mindestens den Namen des angeforderten Dialogs und einen ID-Code.

Wenn Sie die Funktion *'request-dialog* von *'STATUS'* verwenden, lautet der Standardstartname **`CALCULATOR`**.

Es basiert auf PyQts *QDialog*.

Optionen für Einstellungsdatei (engl. preferences file)

Im Abschnitt `CALCULATOR` der Einstellungsdatei können die folgenden Optionen festgelegt werden:

- `constValuesList` – Eine durch Kommas getrennte Liste von häufig verwendeten Werten, die in einer eigenen Button-Reihe am unteren Rand des Taschenrechners angezeigt werden. Zum Beispiel erzeugt die Einstellung `0.100, -0.100` zwei Buttons für +0.100 und -0.100, die beim Kantensuchen auf Zoll-Fräsen häufig verwendet werden. Es können bis zu sechs (6) Werte eingegeben werden; darüber hinaus wird die Liste abgeschnitten. Die Werte müssen gültige Fließkomma- oder Ganzzahlen sein.
- `onShowBehavior` – Eine Liste optionaler Verhaltensweisen, die ausgelöst werden, wenn der Taschenrechner-Dialog angezeigt wird. Jede Option muss durch ein Komma getrennt werden.
  - `CLEAR_ALL` – Führt bei jedem Anzeigen des Taschenrechners ein **Alle löschen** aus. Dadurch werden zuvor eingegebene Werte der letzten Nutzung gelöscht, und der Anzeige-Wert wird auf `0` gesetzt.
  - `FORCE_FOCUS` – Erzwingt den Fokus auf das Eingabefeld des Taschenrechners, sobald das Widget angezeigt wird. Dadurch kann eine physische Tastatur direkt Eingaben liefern, ohne dass zusätzliche Klicks nötig sind. Zudem hat dies den Nebeneffekt, dass der aktuelle Wert markiert wird; Eingaben über die Tastatur ersetzen somit den bestehenden Wert, sofern die

Textmarkierung nicht geändert wird.

- `acceptOnReturnKey` – Wenn auf `True` gesetzt, akzeptiert der Taschenrechner den aktuellen Wert und schließt den Dialog, sobald die Return-/Enter-Taste der Tastatur gedrückt wird. Ist es auf `False` gesetzt, wird die Return-Taste ignoriert und die **Apply**-Schaltfläche muss angeklickt werden. Falls die Schaltfläche **Apply Next** verfügbar ist, führt die Return-Taste diese Aktion aus, während der Dialog geöffnet bleibt.

Untertstützung physischer Tastaturen

Obwohl dieses Widget primär eine touchscreen-freundliche Oberfläche bereitstellen soll, ist es möglich, Werte über eine physische Tastatur einzugeben, typischerweise über den numerischen Ziffernblock. Die Tasten funktionieren größtenteils wie erwartet, es existieren jedoch einige spezielle Tastenfunktionen:

- Die **Enter**- oder **Return**-Taste entspricht dem **Gleichheitszeichen** (=) Operator, wenn eine Berechnung aussteht. Andernfalls wird die **Übernehmen**-Funktion ausgeführt, sofern diese über die `acceptOnReturnKey`-Einstellung aktiviert ist.
- Die **Minus-Taste** (-) ändert das Vorzeichen der aktuell im Rechner angezeigten Zahl, wenn sie zweimal hintereinander gedrückt wird. Wird sie nur einmal gedrückt, führt sie wie gewohnt die Subtraktion aus.
- **Alt+Pfeil nach links** führt die Funktion **Zurück** aus und wechselt – sofern unterstützt – in das vorherige Feld.
- **Alt+Rechter Pfeil** führt die Funktion **Weiter** aus und wechselt zum nächsten Feld, sofern dies unterstützt wird.
- **Alt+Rücktaste** bricht den Rechner ab und gibt keinen Rückgabewert an das aufrufende Widget zurück.

`RunFromLine` - Ausführen-ab-Zeile Dialog-Widget



Figure 325. QtVCP `RunFromLine`: Run-From-Line-Dialog-Widget

Dialog zum **Voreinstellen der Spindeleinstellungen vor dem Ausführen eines Programms in einer bestimmten Zeile**.

Es basiert auf PyQts *QDialog*.

`VersaProbeDialog` - Dialogfeld-Widget für die Berührungsprüfung von Bauteilen

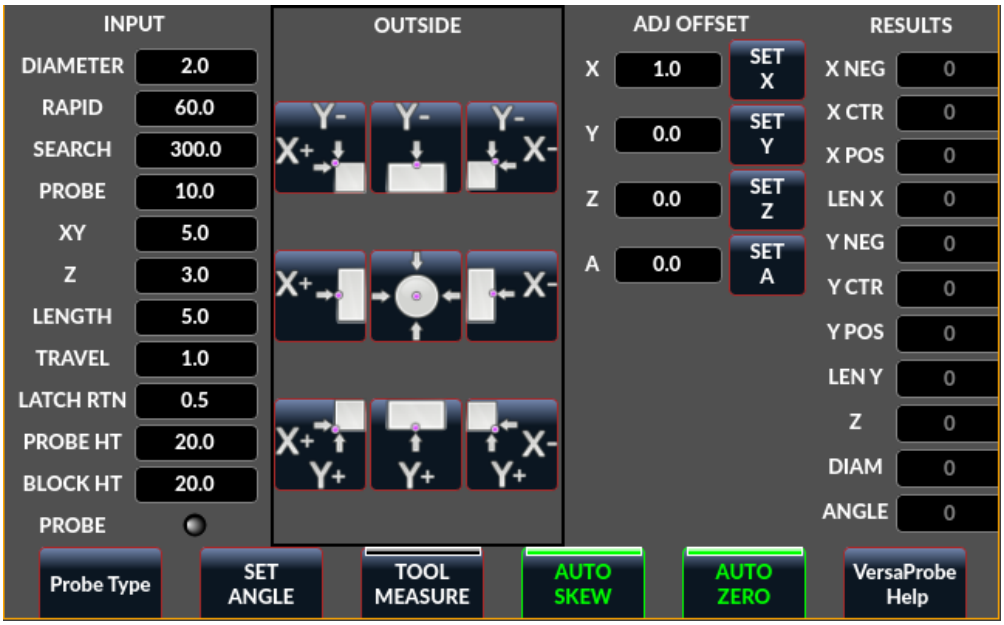


Figure 326. QtVCP `VersaProbeDialog`: Dialog-Widget für Werkstück Touch Probing

Dies ist ein Dialog zur Anzeige eines **Werkstück-Suchbildschirms auf der Basis von Verser Probe v2**.

Es basiert auf PyQts *QDialog*.

`MachineLogDialog` - Maschinen- und Debugging-Protokolldialog-Widget

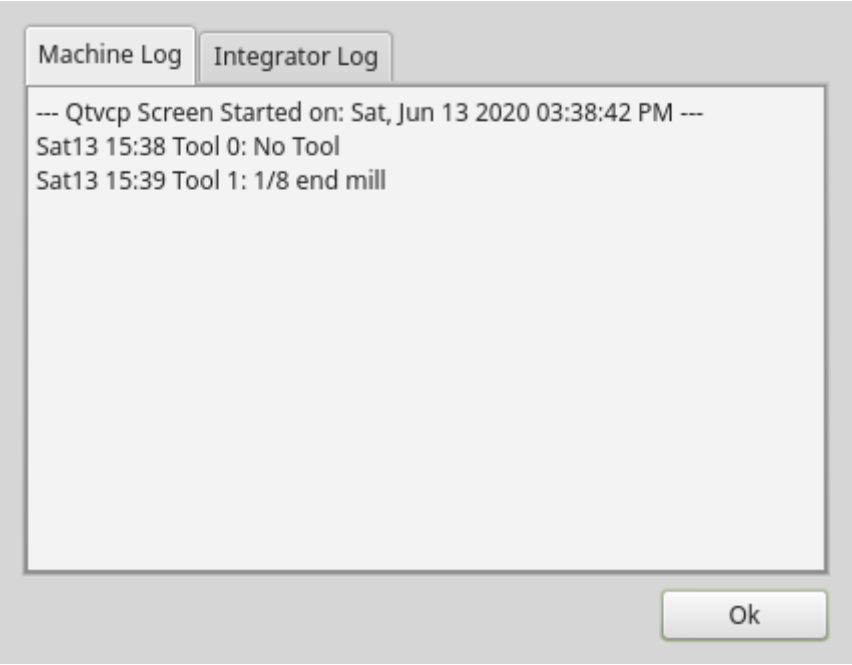


Figure 327. QtVCP `MachineLogDialog`: Dialog-Widget für Maschinen- und Debugging-Protokolle

Dies ist ein Dialog zur **Anzeige des Maschinen-Logfiles und des QtVCPs Debugging Logs**.

Es basiert auf PyQts *QDialog*.

12.8.4. Andere Widgets

Weitere verfügbare Widgets:

`NurbsEditor` - NURBS-Bearbeitungs-Widget

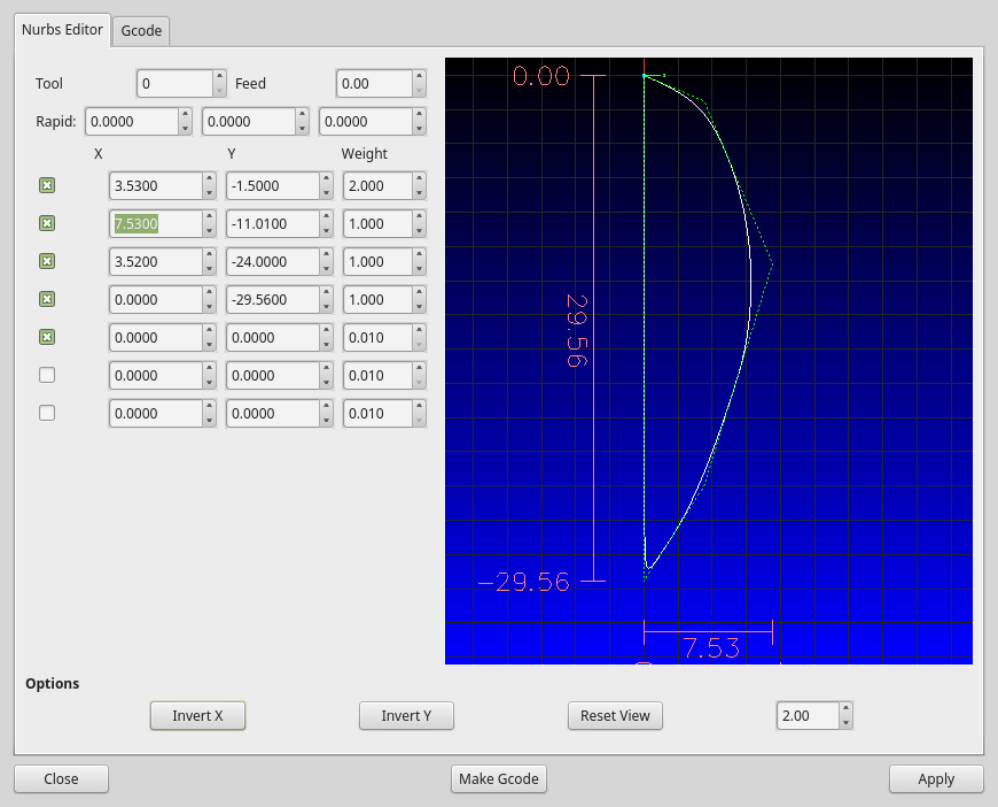


Figure 328. QtVCP `NurbsEditor`: NURBS-Bearbeitungs-Widget

Mit dem Nurbs-Editor können Sie eine NURBS-basierte Geometrie auf dem Bildschirm **manipulieren** und anschließend **NURBS in G-Code konvertieren**.

Sie können den G-Code am Bildschirm bearbeiten und ihn dann an LinuxCNC senden.

Es basiert auf PyQts *QDialog*.

`Joypad` - 5-Tasten-D-Pad-Widget

Es ist die Basisklasse für das `HALPad`-Widget.

Dieses Widget sieht aus und funktioniert wie ein **5-Tasten-D-Pad, mit einer LED-Anzeige in einem Ring**.

Sie können jede der Schaltflächenpositionen mit Text oder Symbolen versehen.

Sie können sich *mit Ausgangssignalen verbinden*, wenn die Tasten gedrückt werden.

Es gibt auch *Eingabe-Slots*, um die Farbe des Indikators(s.) zu ändern

ENUMS

Es gibt **aufgezählte Konstanten, die zur Referenzierung von Indikatorpositionen verwendet werden**.
Sie werden im Eigenschaftseditor des Qt Designer-Editors oder im Python-Code verwendet.

NONE

LEFT, **L**

RIGHT, **R**

CENTER, **C**

TOP, **T**

BOTTOM, **B**

LEFTRIGHT, **X**

TOPBOTTOM, **A**

Für Python-Handler-Code verwenden Sie den Widget-Namen in Qt Designer plus die Referenzkonstante:

```
self.w.joypadname.set_highlight(self.w.joypadname.LEFT)
```

Nützliche Funktionen, die überschrieben werden können

```
def _pressedOutput(self, btncode):
    self.joy_btn_pressed.emit(btncode)
    self[''.format(btncode.lower())].emit(True)

def _releasedOutput(self, btncode):
    self.joy_btn_released.emit(btncode)
    self['joy_{}_pressed'.format(btncode.lower())].emit(False)
```

Wie kodiert, geben diese Funktionen *PyQt5-Signale (joy_btn_pressed und joy*<letter>_pressed) für jeden gedrückten oder losgelassenen Knopf_ aus.

Signal *joy_btn_pressed* gibt einen String-Code für die Taste aus.
Signal *joy_<letter>_pressed* gibt einen Bool-Wert aus.

Sie können die Funktionen überschreiben, um etwas anderes zu implementieren, indem Sie ein benutzerdefiniertes Widget erstellen:

Aufrufbare Funktionen

reset_highlight()

Löscht die Hervorhebungsanzeige.

set_highlight(_button_, state=_True_)

Setzen Sie den Hervorhebungsanzeiger an der Position `button` auf den Zustand `state`.
Sie können *Strings Buchstaben* (`LRCTBXA`) oder *Position* `ENUMS` für das Argument der Schaltfläche verwenden.

set_button_icon(_button_, _pixmap_)

Legt die Pixmap für das Symbol der Schaltfläche fest.

set_button_text(_button_, _text_)

Legt den Text für das Symbol der Schaltfläche fest.

set_tooltip(_button_, _text_)

Legt den beschreibenden Text für die Popup-Tooltip-Schaltflächen fest.

setLight(_state_)

Setzt den Highlight-Indikator auf die Farbe `True` oder `False`.
Die Funktion `set_highlight()` muss vorher verwendet werden, um den zu verwendenden Indikator zu setzen.

Signale

Diese Signale werden **gesendet, wenn Tasten gedrückt werden**.

Sie können im Qt Designer-Editor oder im Python-Code verbunden werden.

Die ersten beiden geben eine Zeichenfolge aus zur Angabe der gedrückten Schaltfläche:

joy_btn_pressed ***(string)***
joy_btn_released ***(string)***
joy_l_pressed ***(bool)***
joy_l_released ***(bool)***
joy_r_pressed ***(bool)***
joy_r_released ***(bool)***
joy_c_pressed ***(bool)***
joy_c_released ***(bool)***
joy_t_pressed ***(bool)***
joy_t_released ***(bool)***
joy_b_pressed ***(bool)***
joy_b_released ***(bool)***

Sie basieren auf PyQt's *Signal* (`QtCore.pyqtSignal()`)

Steckplätze (engl. slots)

Slots können im Qt Designer-Editor oder in Python-Code verbunden werden:

set_colorStateTrue()
set_colorStateFalse()
set_colorState(_bool_)
set_true_color(str)
set_true_color(_qcolor_)
set_false_color(_str_)
set_false_color(_qcolor_)

Eigenschaften

Diese können in Stylesheets oder Python-Code festgelegt werden:

highlightPosition

Position des Indikators festlegen.

setColorState

Den Farbzustand des Indikators auswählen.

left_image_path

right_image_path

center_image_path

top_image_path

bottom_image_path

Ein Dateipfad oder Ressourcenpfad zu einem Bild, das an der beschriebenen Schaltflächenposition angezeigt werden soll.
Wenn die Schaltfläche "Zurücksetzen" in der Qt Designer-Editor-Eigenschaft gedrückt wird, dann wird das Bild nicht angezeigt (optional kann Text angezeigt werden).

left_text

right_text

center_text

top_text

bottom_text

Eine Textzeichenfolge, die an der beschriebenen Schaltflächenposition angezeigt werden soll.
Wenn das Feld leer gelassen wird, kann ein Bild für die Anzeige bestimmt werden.

true_color

false_color

Farbauswahl für den mittleren LED-Ring, der angezeigt werden soll, wenn der `BASENAME.light.center` *HAL-Pin* `True` oder `False` ist.

text_color

Auswahl der Farbe für den Text des Buttons.

button_font

Auswahl der Schriftart für den Text des Buttons.

Die obigen Eigenschaften könnten gesetzt werden in:

- **Stylesheets**:
  Normalerweise würden Sie den Qt Designer Widget-Namen mit `#` *Präfix* verwenden, um individuelle Widget-Eigenschaften zu setzen, andernfalls würden Sie den `JoyPad` *Klassenname* nutzen, um alle `JoyPad` Widgets gleich zu setzen:

```
#joypadname{
  qproperty-true_color: #000;
```

```
  qproperty-false_color: #444;
}
```

- **Im Python-Handler-Code**:

```
self.w.joypadename.setProperty('true_color','green')
self.w.joypadename.setProperty('false_color','red')
```

WebWidget

Dieses Widget erstellt eine html/pdf-Anzeigeseite mit den QtWebKit- oder QtWebEngine-Bibliotheken. Der neuere QtWebEngine wird bevorzugt, wenn beide auf dem System sind.
Wird die QtWebEngine-Bibliothek mit dem Qt Designer-Editor verwendet, wird in Qesigner ein Platzhalter QWidget angezeigt. Dies wird durch das gelaufene QtWebEngine Widget ersetzt.

12.8.5. BaseClass/Mixin-Widgets

Diese Widgets werden verwendet, um **verschiedene Eigenschaften und Verhaltensweisen in anderen Widgets zu kombinieren**.

Sie werden als ausklappbare Kopfzeile in der Eigenschaftsspalte von Qt Designer angezeigt.

`IndicatedPushButtons`

Diese Klasse **verändert das Verhalten von `QPushButton`**.

LED-Anzeige Option

`indicator_option` *setzt eine LED auf die Oberseite der Taste.*



Figure 329. QtVCP `PushButton`: IndicatedPushButton-Schaltfläche, LED-Anzeige-Option

Es kann ein *Dreieck*, *Kreis*, *obere Leiste* oder *seitliche Leiste* sein.
Die Dreieck- und Kreis-LEDs können optional doppelte vertikale LEDs anzeigen, jeweils mit eigenem HAL-Pin. Die *Größe* und *Position* können angepasst werden.

Es wird angezeigt:

- den **aktuellen Zustand der Schaltfläche**, oder
- den **Zustand eines HAL-Pins**, oder
- **LinuxCNC-Status**.

Eigenschaften

Diese Eigenschaften sind verfügbar, um den Indikator anzupassen (nicht alle sind auf jede LED-Form anwendbar):

doubleIndicator

Bei dreieckigen oder runden LEDs fügen Sie eine zweite vertikale LED hinzu.

on_color

off_color

flashIndicator

Wenn der Indikator true ist, so blinke die LED an und aus.

flashRate

Rate des Blinkens

indicator_size

Größe der Dreiecks-LED

circle_diameter

Durchmesser der runden LED

shape_option

0-4 LED Form-Typen

right_edge_offset

Platz von der rechten Kante

top_edge_offset

Platz von der oberen Kante

height_fraction

Verwendet für Bar LEDs

width_fraction

Verwendet für Bar LEDs

corner_radius

Indikator-Eckenradius.

Die Farbe der LED-Anzeige kann in einem *stylesheet* definiert werden, indem der folgende Code zur `.qss`-Datei hinzugefügt wird:

```
Indicated_PushButton{
  qproperty-on_color: #000;
  qproperty-off_color: #444;
}
```

Oder für einen bestimmten Button:

```
Indicated_PushButton #button_estop{
  qproperty-on_color: black;
  qproperty-off_color: yellow;
}
```

Optionen

`IndicatedPushButton` hat **exklusive Optionen**:

`indicator_HAL_pin_option`

Fügt ein `halpin` namens `<buttonname>-led` hinzu, der den Status der Schaltflächenanzeige steuert.

`indicator_status_option`

Lässt die LED den Status dieser wählbaren LinuxCNC-Status anzeigen:

- *Is Estopped*
- *Is On*
- *All Homed*
- *Is Joint Homed*
- *Idle*
- *Paused*
- *Flood*
- *Mist* (engl. Nebel)
- *Block Delete*
- *Optionaler Stop*
- *Manual*
- *MDI*
- *Auto*
- *Spindle Stopped*
- *Spindel vorwärts*
- *Spindel rückwärts*
- *On Limits*

Einige `indicator_status_options` enthält eine Eigenschaft, die mit einem *stylesheet* verwendet werden kann, um die Farbe der Schaltfläche basierend auf dem Zustand der Eigenschaft in LinuxCNC zu ändern.
Derzeit sind diese Status-Eigenschaften können verwendet werden, um Auto-Stil Schaltflächen:

- `is_estopped_status` schaltet die Eigenschaft `isEstop` um
- `is_on_status` schaltet die Eigenschaft `isStateOn` um
- `is_manual_status`, `is_mdi_status`, `is_auto_status` schalten die Eigenschaften `isManual`,

`isMDI`, und `isAuto` um.

- `is_homed_status` schaltet die Eigenschaft *isAllHomed* um

Hier ist ein Beispiel-Stylesheet-Eintrag, der den Hintergrund von Mode-Button-Widgets festlegt, wenn sich LinuxCNC in diesem Modus befindet:

```
ActionButton[isManual=true] {
    background: red;
}
ActionButton[isMdi=true] {
    background: blue;
}
ActionButton[isAuto=true] {
    background: green;
}
```

So geben Sie ein bestimmtes Widget anhand seines objectName in Qt Designer an:

```
ActionButton #estop button [isEstopped=false] {
    color: yellow;
}
```

Aktiviert durch den LinuxCNC-Status

Oft, mit der Schaltfläche deaktiviert und aktiviert auf der Grundlage der Zustand der LinuxCNC Motion Controller ist notwendig.

Es gibt mehrere Eigenschaften, die zur Unterstützung ausgewählt werden können:

`isAllHomedSentive`

`isOnSensitive`

`isIdleSensitive`

`isRunSensitive`

`isRunPausedSensitive`

`isManSensitive`

`isMDISensitive`

`isAutoSensitive`

Sie können mehrere Eigenschaften für kombinierte Anforderungen auswählen.

Text ändert sich im Zustand

Die Auswahl der Option **`checked_state_text_option`** erlaubt es einer *checkbox, den Text basierend auf ihrem checked state zu ändern.*

Es verwendet die folgenden Eigenschaften, um den Text für jeden Zustand anzugeben:

true_state_string

false_state_string

`\\n` wird in einen Zeilenumbruch umgewandelt.

Sie können diese in Stylesheets festlegen/ändern:

```
ActionButton #action_aux{
  qproperty-true_state_string: "Air\\nOn";
  qproperty-false_state_string: "Air\\nOff";
}
```

Python-Befehle für den Zustand aufrufen

Die **python_command_option** ermöglicht es, kleine Schnipsel von Python-Code auf Knopfdruck auszuführen, ohne die Handler-Datei bearbeiten zu müssen. Es kann jedoch Funktionen in der Handler-Datei aufrufen.

Bei Verwendung der `command_string`-Eigenschaften.

true_python_cmd_string

Ein Python-Befehl, der aufgerufen wird, wenn der Button auf `True` umgeschaltet wird.

false_python_cmd_string

Ein Python-Befehl, der aufgerufen wird, wenn die Schaltfläche auf `False` umgeschaltet wird.

Besondere Wörter in Großbuchstaben geben Zugang zu den folgenden Informationen:

INSTANCE

Ermöglicht den Zugriff auf die Instanzen der Widgets und die Handler-Funktionen.
Z.B.: `INSTANCE.my_handler_function_call(True)`

ACTION

Ermöglicht den Zugriff auf die `ACTION` Bibliothek von QtVCP.
Z.B. `ACTION.TOGGLE_FLOOD()`

PROGRAM_LOADER

Ermöglicht den Zugriff auf die `PROGRAM_LOADER` Bibliothek von QtVCP.
Z.B., `PROGRAM_LOADER.load_halshow()`

HAL

Ermöglicht den Zugriff auf das Python-Modul von HAL.
Z.B.: `HAL.set_p('motion.probe-input','1')`

12.8.6. Nur-Import-Widgets

Diese Widgets sind normalerweise die **Basisklasse Widget für andere QtVCP-Widgets**.

Sie sind *nicht direkt im Qt-Designer-Editor verfügbar*, können aber **importiert und manuell eingefügt**

werden.

Sie könnten auch **unterklassifiziert** werden, um ein ähnliches Widget mit neuen Funktionen zu erstellen.

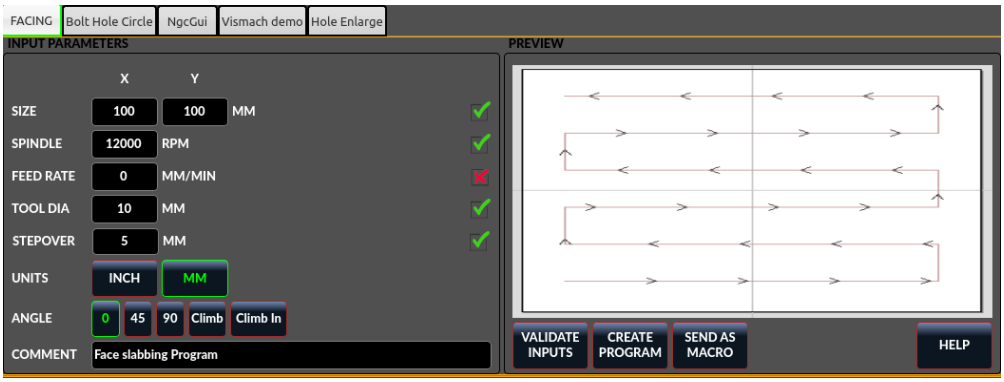
Automatische Höhe

Widget zur Messung von zwei Höhen mit einer Sonde.
Für die Einrichtung.

G-Code Dienstprogramm

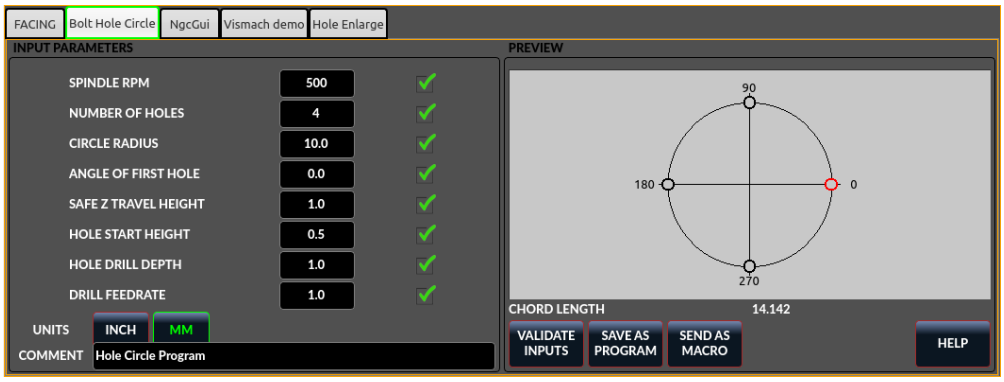
Widgets für die Durchführung gängiger Bearbeitungsprozesse.

Facing



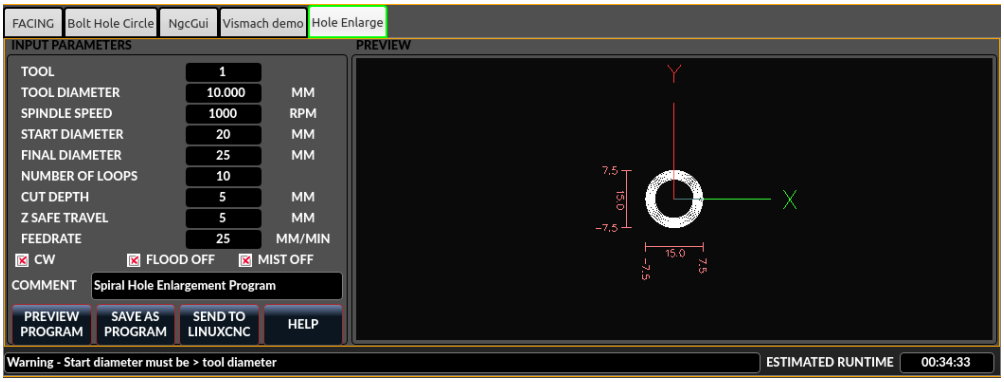
Eine Fläche oder ein definierter Bereich mit unterschiedlichen Strategien bearbeiten, auch *facen*.

Loch-Kreis (engl. hole circle)



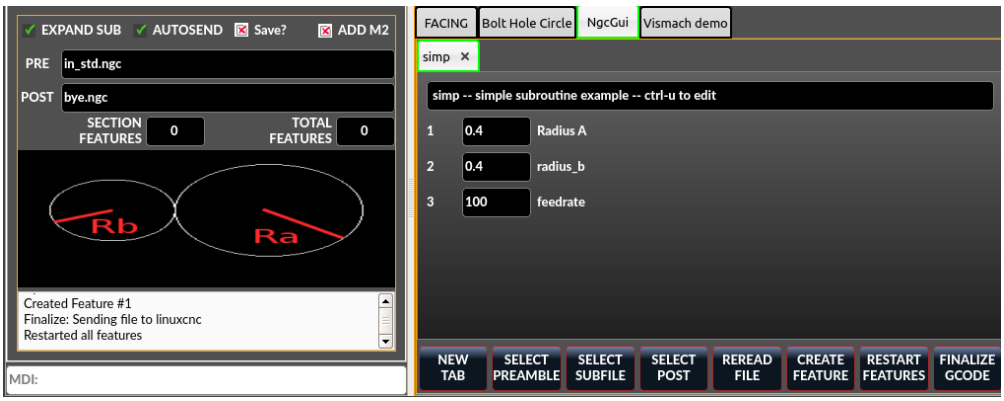
Bohren mehrerer Löcher auf einem Lochkreis.

Loch vergrößern



Nutzt einen Fräser, um ein gebohrtes Loch zu vergrößern.

Qt NGCGUI



QtVCPs Version des NGC-Unterprogramm-Selektors (Gezeigt wie in QtDragon verwendet).

INI-Einstellungen

LinuxCNC muss wissen, wo es nachschauen muss, um die Unterprogramme auszuführen.
Wenn das Unterprogramm andere Unterprogramme oder benutzerdefinierte M-Codes aufruft, müssen diese Pfade ebenfalls hinzugefügt werden.

```
[RS274NGC]
SUBROUTINE_PATH =
~/linuxcnc/nc_files/examples/ngcgui_lib:~/linuxcnc/nc_files/examples/ngcgui_lib/utilitysu
bs
```

QtVCP muss wissen, von wo aus Unterprogramme geöffnet werden sollen.
Sie können auch Unterprogramme angeben, die in Tabs vorgeöffnet werden sollen.

```
[DISPLAY]
# NGCGUI Unterprogramm Pfad.
# Dieser Pfad muss auch in [RS274NGC] SUBROUTINE_PATH stehen
NGCGUI_SUBFILE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib
# vorgewählte Programme Tabs
# nur Dateinamen angeben, Dateien müssen sich im NGCGUI_SUBFILE_PATH befinden
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

Buttons

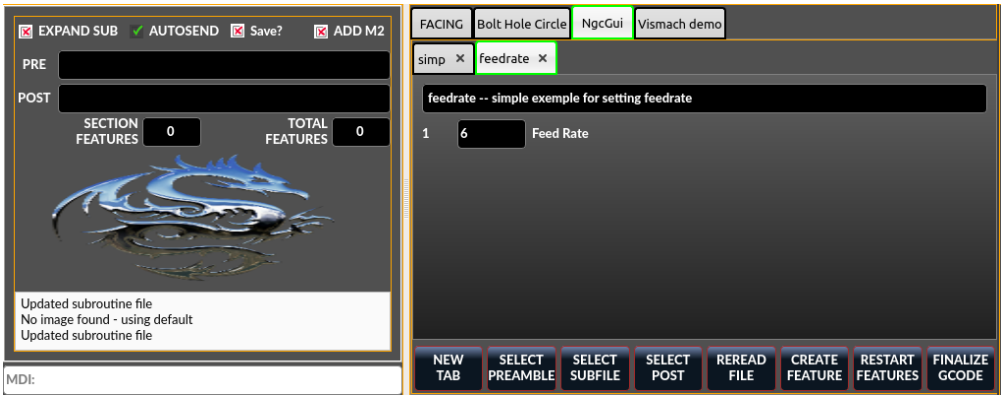
- *NEW TAB* - fügt eine neue leere Registerkarte zu NGCGUI hinzu
- *SELECT PREAMBLE* - Wählen Sie eine Datei aus, die Präambel-G-Code enthält
- *SELECT SUBFILE* - Wählen Sie eine NGCGUI-Unterprogrammdatei
- *SELECT POST* - Wählen Sie eine Datei, die Post-G-Code hinzufügt
- *REREAD FILE* - Erneutes Laden einer Unterprogrammdatei
- *CREATE FEATURE* - Merkmal zur Liste hinzufügen
- *RESTART FEATURE* - alle Features aus der Liste entfernen
- *FINALIZE GCODE* - Erstellen Sie den vollständigen G-Code und senden Sie ihn an LinuxCNC/an eine Datei

Hinzufügen benutzerdefinierter Unterroutinen

Sie können Ihre eigenen Unterroutinen zur Verwendung mit NGCGUI erstellen. Sie müssen diese Regeln befolgen:

- Um ein Unterprogramm für die Verwendung mit NGCGUI zu erstellen, müssen der Dateiname und der Name des Unterprogramms identisch sein.
- Das Unterprogramm muss sich in einem Ordner innerhalb des INI-Suchpfades von LinuxCNC befinden.
- In der ersten Zeile kann ein Kommentar des Typs info stehen:
- Das Unterprogramm muss von den Tags `sub` und `endsub` umgeben sein.
- Die verwendeten Variablen müssen nummerierte Variablen sein und dürfen keine Nummer überspringen.
- Kommentare und Voreinstellungen können enthalten sein.
- Wenn sich eine Bilddatei mit demselben Namen in dem Ordner befindet, wird sie angezeigt.

```
(info: feedrate -- einfaches Beispiel für die Einstellung der Vorschubgeschwindigkeit)
o<feedrate> sub
    #<feedrate> = #1 (= 6 Feed Rate) ; Kommentare in Klammern werden in ngcui angezeigt
    f#<feedrate>
o<feedrate> endsub
```



Qt PDF

Ermöglicht das Hinzufügen von ladbaren PDFs zu einem Bildschirm.

Qt Vismach

Verwenden Sie dies, um OpenGl simulierte Maschinen zu erstellen/hinzuzufügen.

Hal Auswahl Box

Dieses Widget ist eine Combobox zur Auswahl eines Pins oder Signals auf dem System.

```
from qtvcp.widgets.hal_selectionbox import HALSelectionBox

    def buildComboBox(self):
        # combo box for HAL pin selection
        combobox = HALSelectionBox()
        combobox.setShowTypes([combobox.PINS,combobox.SIGNALS])
        combobox.setPinTypes([combobox.HAL_BOOL], direction = [combobox.HAL_IN])
        combobox.setSignalTypes([combobox.HAL_BOOL], driven = [False,True])
        combobox.hal_init()
        combobox.selectionUpdated.connect(lambda w: self.signalSelected(w))

    def signalSelected(self, sig):
        print('Watching:',sig)
```

Dies sind Funktionsaufrufe

```
# setzt die Liste der anzuzeigenden Elemente: PINS SIGNALS
combobox.setShowTypes([combobox.PINS])

# set the pin types to show: HAL_BOOL,HAL_REAL,HAL_SINT,HAL_UINT
# and a list of directions: HAL_IN HAL_OUT
combobox.setPinTypes(types=[combobox.HAL_BOOL], direction = [HAL_IN])

# set the signal types to show: HAL_BOOL,HAL_REAL,HAL_SINT,HAL_UINT
# and a list of driven/undriven (by a connected pin) to show
combobox.setSignalTypes( types=[combobox.HAL_BOOL], driven = [True,True])
```

12.9. QtVCP-Bibliotheksmodule

Bibliotheken sind vorgefertigte Python-Module, die GladeVCP zusätzliche Funktionen verleihen. Auf diese Weise können Sie auswählen, welche Funktionen Sie wünschen - und müssen die üblichen nicht selbst erstellen.

12.9.1. `Status`

`Status` ist eine Bibliothek, die **GObject Nachrichten basierend auf LinuxCNCs aktuellen Status** sendet. Es ist eine *Erweiterung von GladeVCP's GStat Objekt.*

Es hat auch einige Funktionen, um den Status von Dingen wie der internen Jog-Rate zu melden.

Sie *verbinden einen Funktionsaufruf* mit der `STATUS`-Nachricht, an der Sie interessiert sind, und QtVCP wird diese Funktion aufrufen, wenn die Nachricht von `STATUS` gesendet wird.

Anwendung

- **Importiere `Status` Module** +
  Fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Status
```

- **Instantiierung des Moduls `Status`** +
  Fügen Sie diesen Python-Code in Ihren "instantiate"-Abschnitt ein:

```
STATUS = Status()
```

- **Verbindung zu `STATUS`-Meldungen** +
  Verwenden Sie die *GObject-Syntax*.

Beispiel

So können Sie z. B. Ein- und Ausschaltmeldungen der Maschine auffangen.

NOTE

Das folgende Beispiel zeigt *zwei gängige Möglichkeiten, Signale zu verbinden*, eine davon *unter Verwendung von Lambda*. `lambda` wird *verwendet, um Argumente aus der Statusmeldung zu entfernen oder zu manipulieren*, bevor die Funktion aufgerufen wird. Sie können den Unterschied in der Signatur der aufgerufenen Funktion sehen: Diejenige, die Lambda verwendet, akzeptiert das Statusobjekt nicht - Lambda hat es nicht an die Funktion übergeben.

- Fügen Sie diese Befehle in den Abschnitt `[INITIALIZE]` der Python-Handler-Datei ein:

```
STATUS.connect('state-on', self.on_state_on)
STATUS.connect('state-off', lambda: w, self.on_state_off())
```

  Wenn sich LinuxCNC in *Zustand "machine on" befindet, wird in diesem Beispielcode state* die Funktion *self.on_state_on* aufgerufen.
  Wenn LinuxCNC in *Zustand "machine off"* ist, wird die Funktion *self.on_state_off* aufgerufen.

- Diese würden Funktionen aufrufen, die wie diese aussehen:

```
def on_state_on(self, status_object):
    print('LinuxCNC Maschine ist eingeschaltet')
```

```
def on_state_off(self):
    print('LinuxCNC-Maschine ist aus')
```

12.9.2. Info

Info ist eine Bibliothek zum **Sammeln und Filtern von Daten aus der INI-Datei**.

Verfügbare Daten und Voreinstellungen

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")

IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, "share","qtvcp")

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = self.INI.getint("KINS","JOINTS", fallback=0)
AVAILABLE_AXES = ['X','Y','Z']
AVAILABLE_JOINTS = [0,1,2]
GET_NAME_FROM_JOINT = {0:'X',1:'Y',2:'Z'}
GET_JOG_FROM_NAME = {'X':0,'Y':1,'Z':2}
NO_HOME_REQUIRED = False
HOME_ALL_FLAG
JOINT_TYPE = self.INI.getstring(section, "TYPE", fallback="LINEAR")
JOINT_SEQUENCE_LIST
JOINT_SYNC_LIST

JOG_INCREMENTS = None
ANGULAR_INCREMENTS = None
GRID_INCREMENTS

DEFAULT_LINEAR_JOG_VEL = 15 Einheiten pro Minute
MIN_LINEAR_JOG_VEL = 60 Einheiten pro Minute
Länge_LINEAR_JOG_VEL = 300 Einheiten pro Minute

DEFAULT_ANGULAR_JOG_VEL =
MIN_ANGULAR_JOG_VEL =
```

```
MAX_ANGULAR_JOG_VEL =

MAX_FEED_OVERRIDE =
MAX_TRAJ_VELOCITY =

AVAILABLE_SPINDLES = self.INI.getint("TRAJ", "SPINDLES", fallback=1)
DEFAULT_SPINDLE_0_SPEED = 200
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_0_OVERRIDE = 100
MIN_SPINDLE_0_OVERRIDE = 50

MAX_FEED_OVERRIDE = 1.5
MAX_TRAJ_VELOCITY
```

Dialogfenster für Benutzernachrichten

```
USRMESS_BOLDTEXT = self.INI.findall("DISPLAY", "MESSAGE_BOLDTEXT")
USRMESS_TEXT = self.INI.findall("DISPLAY", "MESSAGE_TEXT")
USRMESS_TYPE = self.INI.findall("DISPLAY", "MESSAGE_TYPE")
USRMESS_PINNAME = self.INI.findall("DISPLAY", "MESSAGE_PINNAME")
USRMESS_DETAILS = self.INI.findall("DISPLAY", "MESSAGE_DETAILS")
USRMESS_ICON = self.INI.findall("DISPLAY", "MESSAGE_ICON")
ZIPPED_USRMESS =

self.GLADEVCP = self.INI.find("DISPLAY", "GLADEVCP")
```

Eingebettete Programminformationen

```
TAB_NAMES = (self.INI.findall("DISPLAY", "EMBED_TAB_NAME")) or None
TAB_LOCATION = (self.INI.findall("DISPLAY", "EMBED_TAB_LOCATION")) or []
TAB_CMD = (self.INI.findall("DISPLAY", "EMBED_TAB_COMMAND")) or None
ZIPPED_TABS =

MDI_COMMAND_LIST =      (heading: [MDI_COMMAND_LIST], title: MDI_COMMAND")
TOOL_FILE_PATH =        (heading: [EMCIO], title:TOOL_TABLE)
POSTGUI_HALFILE_PATH =  (heading: [HAL], title: POSTGUI_HALFILE)
```

Helfer

Es gibt einige *Hilfsfunktionen*, die hauptsächlich zur Unterstützung von Widgets verwendet werden:

get_error_safe_setting(_self_, _heading_, _detail_, default=_None_)

convert_metric_to_machine(_data_)

convert_imperial_to_machine(_data_)

convert_9_metric_to_machine(_data_)

convert_9_imperial_to_machine(_data_)

convert_units(_data_)

convert_units_9(_data_)

get_filter_program(_fname_)

get_qt_filter_extensions()

Filtererweiterungen im Qt-Format abrufen.

Anwendung

- **Importiere `Info` Modul**
  Fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Info
```

- **Instantiierung des Moduls `Info`.**
  Fügen Sie diesen Python-Code in Ihren "instantiate"-Abschnitt ein:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

INFO = Info()
```

- **Zugriff auf `INFO`-Daten**. Verwenden Sie diese allgemeine Syntax:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print('Metric based')
```

12.9.3. `Action`

`Action` Bibliothek wird verwendet, um **die Bewegungssteuerung von LinuxCNC** zu steuern.

Es versucht, zufällige Details zu verbergen und praktische Methoden für Entwickler hinzuzufügen.

Helfer

Es gibt einige **Hilfsfunktionen**, die hauptsächlich für die Unterstützung dieser Bibliothek verwendet

werden:

`get_jog_info (_num_)`

`jnum_check(_num_)`

`ensure_mode(_modes_)`

`open_filter_program(_filename_, _filter_)`

Öffnen Sie das G-Code-Filterprogramm.

Anwendung

- **Importiere `Action` Modul**
  Fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Action
```

- **Instantiierung des Moduls `Action`**
  Fügen Sie diesen Python-Code in Ihren instantiate-Abschnitt ein:

```
###########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
###########################################

ACTION = Action()
```

- **Zugriff auf `ACTION`-Befehle**
  Verwenden Sie eine allgemeine Syntax wie die folgende:

```
ACTION.SET_ESTOP_STATE(state)
ACTION.SET_MACHINE_STATE(state)

ACTION.SET_MACHINE_HOMING(joint)
ACTION.SET_MACHINE_UNHOMED(joint)

ACTION.SET_LIMITS_OVERRIDE()

ACTION.SET_MDI_MODE()
ACTION.SET_MANUAL_MODE()
ACTION.SET_AUTO_MODE()

ACTION.SET_LIMITS_OVERRIDE()

ACTION.CALL_MDI(code)
ACTION.CALL_MDI_WAIT(code)
ACTION.CALL_INI_MDI(number)

ACTION.CALL_OWORD()
```

```
ACTION.OPEN_PROGRAM(filename)
ACTION.SAVE_PROGRAM(text_source, fname):

ACTION.SET_AXIS_ORIGIN(axis,value)
ACTION.SET_TOOL_OFFSET(axis,value,fixture = False)

ACTION.RUN()
ACTION.ABORT()
ACTION.PAUSE()          # Wechselt zwischen Pause/Fortsetzen
ACTION.PAUSE_MACHINE()
ACTION.RESUME()

ACTION.SET_MAX_VELOCITY_RATE(rate)
ACTION.SET_RAPID_RATE(rate)
ACTION.SET_FEED_RATE(rate)
ACTION.SET_SPINDLE_RATE(rate)

ACTION.SET_JOG_RATE(rate)
ACTION.SET_JOG_INCR(incr)
ACTION.SET_JOG_RATE_ANGULAR(rate)
ACTION.SET_JOG_INCR_ANGULAR(incr, text)

ACTION.SET_SPINDLE_ROTATION(direction = 1, rpm = 100, number = 0)
ACTION.SET_SPINDLE_FASTER(number = 0)
ACTION.SET_SPINDLE_SLOWER(number = 0)
ACTION.SET_SPINDLE_STOP(number = 0)

ACTION.SET_USER_SYSTEM(system)

ACTION.ZERO_G92_OFFSET()
ACTION.ZERO_ROTATIONAL_OFFSET()
ACTION.ZERO_G5X_OFFSET(num)

ACTION.RECORD_CURRENT_MODE()
ACTION.RESTORE_RECORDED_MODE()

ACTION.SET_SELECTED_AXIS(jointnum)

ACTION.DO_JOG(jointnum, direction)
ACTION.JOG(jointnum, direction, rate, distance=0)

ACTION.TOGGLE_FLOOD()
ACTION.SET_FLOOD_ON()
ACTION.SET_FLOOD_OFF()

ACTION.TOGGLE_MIST()
ACTION.SET_MIST_ON()
ACTION.SET_MIST_OFF()

ACTION.RELOAD_TOOLTABLE()
ACTION.UPDATE_VAR_FILE()

ACTION.TOGGLE_OPTIONAL_STOP()
ACTION.SET_OPTIONAL_STOP_ON()
ACTION.SET_OPTIONAL_STOP_OFF()
```

```
ACTION.TOGGLE_BLOCK_DELETE()
ACTION.SET_BLOCK_DELETE_ON()
ACTION.SET_BLOCK_DELETE_OFF()

ACTION.RELOAD_DISPLAY()
ACTION.SET_GRAPHICS_VIEW(view)

ACTION.UPDATE_MACHINE_LOG(text, option=None):

ACTION.CALL_DIALOG(command):

ACTION.HIDE_POINTER(state):

ACTION.PLAY_SOUND(path):
ACTION.PLAY_ERROR():
ACTION.PLAY_DONE():
ACTION.PLAY_READY():
ACTION.PLAY_ATTENTION():
ACTION.PLAY_LOGIN():
ACTION.PLAY_LOGOUT():
ACTION.SPEAK(speech):

ACTION.BEEP():
ACTION.BEEP_RING():
ACTION.BEEP_START():

ACTION.SET_DISPLAY_MESSAGE(string)
ACTION.SET_ERROR_MESSAGE(string)

ACTION.TOUCHPLATE_TOUCHOFF(search_vel, probe_vel, max_probe,
       z_offset, retract_distance, z_safe_travel, rtn_method=None, error_rtn = None)
```

12.9.4. Qhal

Eine Bibliothek für die Interaktion mit HAL-Komponenten/System.

Attribute

Dies sind Funktionen zur Anwendung auf Qhal Objekte:

`setUpdateRate(cyclerate)`

Zyklusrate in ms festlegen

`newPin(Name, Pin-Typ Konstante, Pin-Richtung Konstante)`

Gibt ein neues QPin-Objekt zurück

`getPinObject(name)`

Gibt ein vorhandenes QPin-Objekt mit dem angegebenen Namen zurück

`getValue(name)`

Gibt den Wert des angegebenen Pins zurück. Nutzt den vollständigen Komponenten.pin-Namen.

`setPin(name,value)`

Setzt den Wert des angegebenen Pins. Nutzt den vollständigen Komponenten.Pin-Namen.

`setSignal(name,value)`

Setzt den Wert des angegebenen Signals. Nutzt den vollständigen Komponenten.Pin-Namen.

`makeUniqueName(Name)`

gibt eine Zeichenkette mit eindeutigen HAL Pin Namen zurück durch Hinzufügen von *-x* (einer Nummer) zur gegebenen Zeichenkette eines Pin-Namens

`exit()`

killt (eingedeutschter englischer slang, gemeint ist der Abbruch der Ausführung) die Komponente

Konstanten

Es sind die drei verfügbaren Konstanten:

- **HAL_BOOL**
- **HAL_REAL**
- **HAL_SINT**
- **HAL_UINT**
- **HAL_IN**
- **HAL_OUT**
- **HAL_IO**
- **HAL_RO**
- **HAL_RW**

Referenzen

Verfügbare Objekt-Referenzen:

- **comp** das Komponenten-Objekt
- **hal** das hal Bibliothek object

12.9.5. QPin

Wrapper-Klasse von HAL pins

Signale

Es gibt drei Qt-Signale mit denen der QPin verbunden werden kann:

- **value_changed** wird eine Funktion über ihren Namen aufrufen mit einem Argument des aktuellen Wertes (veraltet)

- * pinValueChanged* wird eine Funktion über ihren Namen aufrufen mit Argumentes des Pin Objektes und des aktuellen Wertes
- * isDrivenChanged* wird eine Funktion über ihren Namen aufrufen mit Argumentes des Pin Objektes und dessen aktuellen Zustands (engl. state) der Pin ist (nicht) verbunden mit einem treibenden (engl. driving) Pin

Attribute

Dies sind die Funktionen, die auf einen QPin angewendet werden können:

- **<Pin object>.get()** gibt den aktuellen Wert des Pin Objektes zurück
- **<Pin Objekt>.set(X)** setzt den Werte des Pin Objektes auf den Wert X
- **<Pin Objekt>.text()** gibt den Pin-Namen als Zeichenkette zurück

Referenzen

Verfügbare Objekt-Referenzen:

- **hal** das hal Bibliothek object

Beispiel

Hinzufügen einer Funktion, die aufgerufen wird, wenn sich der Zustand des Pins ändert

```
from qtvcp.core import Qhal
QHAL = Qhal()

########################################
# Spezielle Funktionen, die von QtVCP aufgerufen werden
########################################

# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    self.pin_button_in = QHAL.newpin('cycle-start-in',QHAL.HAL_BOOL, QHAL.HAL_IN)
    self.pin_button_in.pinValuechanged.connect(self.buttonChanged)
    self.pin_button_in.isDrivenChanged.connect(lambda p,s: self.buttonDriven(p,s))

def buttonChanged(self, pinObject, value):
    print('Pin Name:{} änderte Werte zu {}'.format(pinObject.text(), value))

def buttonDriven(self, pinObject, state):
    message = 'nicht getrieben (engl. driven) durch einen Ausgabe (engl. output)-Pin'
    if state:
        message = 'ist getrieben (engl. driven) durch einen Ausgabe (engl. output)-Pin'
    print('Pin name:{} is {}'.format(pinObject.text(), message))
```

12.9.6. `Tool`

Diese Bibliothek **verarbeitet Änderungen an Werkzeugoffset-Dateien**.

WARNING	* LinuxCNC verarbeitet fremde (engl. third party) Werkzeugdateien nicht gut.*

Helfer

`GET_TOOL_INFO(_toolnumber_)`

Dies liefert eine Python **Liste mit Informationen über die angeforderte Werkzeugnummer**.

`GET_TOOL_ARRAY()`

Dies liefert eine einzelne Python **Liste von Python-Listen mit Werkzeuginformationen**.

Dies ist eine Rohliste, die *aus der Systemtooldatei* gebildet wird.

`ADD_TOOL(_newtool_ = [_-99, 0,'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','New Tool'_])`

Das Ergebnis ist ein Python **Tupel aus zwei Python-Listen von Python-Listen mit Werkzeuginformationen**:

- `[0]` werden *echte Werkzeuginformationen* sein
- `[1]` sind *Verschleißinformationen* (Werkzeugnummern sind über 10000; Fanuc-Stil Werkzeugverschleiß)

Standardmäßig wird ein leerer Werkzeugeintrag mit der Werkzeugnummer -99 hinzugefügt.
Sie können das Array `newtool` mit Werkzeuginformationen vorbereiten.

`DELETE_TOOLS(_toolnumber_)`

Löschen des nummerierten Werkzeugs.

`SAVE_TOOLFILE(_toolarray_)`

Dies wird **das `toolarray` parsen und es in der Werkzeugdatei speichern**, die in der *INI-Datei* als Werkzeugpfad angegeben ist.

Dieses Werkzeug *array muss alle verfügbaren Werkzeuginformationen enthalten*.

Es wird erwartet, dass dieses Array das LinuxCNC *raw-Tool array* verwendet, d.h. es hat keine Werkzeugverschleißeinträge.

Sie gibt True zurück, wenn ein Fehler aufgetreten ist.

`CONVERT_TO_WEAR_TYPE(_toolarray_)`

Diese Funktion **konvertiert ein LinuxCNC raw tool array in ein QtVCP tool array**.

Das Tool-Array von *QtVCP enthält Einträge für den Werkzeugverschleiß der X- und Z-Achse*.

LinuxCNC unterstützt Werkzeugverschleiß, indem es **Werkzeugverschleißinformationen in Werkzeugeinträge über 10000** einfügt.

NOTE	Dies erfordert auch **Remap-Code, um die Verschleißversätze zum Zeitpunkt des Werkzeugwechsels hinzuzufügen**.

`CONVERT_TO_STANDARD_TYPE(_toolarray_)`

Diese Funktion **konvertiert das QtVCP-Werkzeugfeld in ein LinuxCNC-Rohwerkzeugfeld**.

Das Array von *QtVCP enthält Einträge für den Werkzeugverschleiß der X- und Z-Achse*.

LinuxCNC unterstützt Werkzeugverschleiß, indem es **Werkzeugverschleißinformationen in Werkzeugeinträge über 10000** einfügt.

NOTE	Dies erfordert auch **Remap-Code, um die Verschleißkompensationen und Werkzeugwechselzeit hinzuzufügen**.

12.9.7. `Path`

Das Modul `Path` gibt **Verweise auf wichtige Dateipfade**.

Referenzierte Pfade

`PATH.PREFS_FILENAME`

Der Pfad der Einstellungsdatei.

`PATH.WORKINGDIR`

Das Verzeichnis, aus dem QtVCP gestartet wurde.

`PATH.IS_SCREEN`

Ist dies eine Eingabemaske (engl. screen) oder ein VCP?

`PATH.CONFIGPATH`

Gestarteter Konfigurationsordner.

`PATH.RIPCONFIGDIR`

Der Run-in-Place-Konfigurationsordner für QtVCP-Bildschirme.

`PATH.BASEDIR`

Basisordner für LinuxCNC.

`PATH.BASENAME`

Der Name der Qt Designer-Dateien (ohne Endung).

`PATH.IMAGEDIR`

Der QtVCP-Bildordner.

`PATH.SCREENDIR`

Der in QtVCP integrierte Bildschirmordner.

PATH.PANELDIR

Der in QtVCP eingebaute VCP-Ordner.

PATH.HANDLER

Handler-Datei Pfad.

PATH.HANDLERDIR

Verzeichnis, in dem die Python-Handler-Datei gefunden wurde.

PATH.XML

QtVCP UI-Dateipfad.

PATH.HANDLERDIR

Verzeichnis, in dem die UI-Datei gefunden wurde.

PATH.QSS

QtVCP QSS-Dateipfad.

PATH.PYDIR

LinuxCNCs Python-Bibliothek.

PATH.LIBDIR

Der Ordner der QtVCP-Bibliothek.

PATH.WIDGET

Der QtVCP-Widget-Ordner.

PATH.PLUGIN

Der QtVCP-Widget-Plugin-Ordner.

PATH.VISMACHDIR

Verzeichnis, in dem sich die vorgefertigten Vismach-Dateien befinden.

Derzeit nicht verwendet:

PATH.LOCALEDIR

Ordner für Übersetzungen.

PATH.DOMAIN

Übersetzungsbereich (engl. translation domain).

Helfer

Es sind einige Hilfsfunktionen verfügbar:

```
file_list = PATH.find_vismach_files()
directory_list = PATH.find_screen_dirs()
directory_list = PATH.find_panel_dirs()
```

Anwendung

- **Importiere `Path` Modul** +
  Fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Path
```

- **Instantiierung des Moduls `Path`** +
  Fügen Sie diesen Python-Code in Ihren instantiate-Abschnitt ein:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

PATH = Path()
```

12.9.8. `VCPWindow`

Das `VCPWindow` Modul gibt **Referenz auf das `MainWindow` und Widgets**.

Typischerweise wird dies für eine Bibliothek verwendet (z.B. für die Toolbar-Bibliothek), da die Widgets einen Verweis auf das `MainWindow` von der Funktion `_hal_init()` erhalten.

Anwendung

- **Import *VCPWindow* Modul** +
  Fügen Sie diesen Python-Code zu Ihrem Importabschnitt hinzu:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.qt_makegui import VCPWindow
```

- **VCPWindow-Modul instanziieren**
  Fügen Sie diesen Python-Code in Ihren instantiate-Abschnitt ein:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

WIDGETS = VCPWindow()
```

12.9.9. `Aux_program_loader`

`Aux_program_loader` Modul ermöglicht einen einfachen Weg zu **Hilfsprogramme zu laden, die LinuxCNC oft benutzt**.

Helfer

`load_halmeter()`

Halmeter wird zur **Anzeige von Daten eines HAL-Pins** verwendet.
Laden Sie ein `Halmeter` mit:

```
AUX_PRGM.load_halmeter()
```

`load_ladder()`

SPS-Programm *ClassicLadder* laden:

```
AUX_PRGM.load_ladder()
```

`load_status()`

Laden des LinuxCNC `status` Programms:

```
AUX_PRGM.load_status()
```

`load_halshow()`

HALshow laden, Anzeigeprogramm konfigurieren:

```
AUX_PRGM.load_halshow()
```

`load_halscope()`

Laden des *HALscope* Programms:

```
AUX_PRGM.load_halscope()
```

`load_tooledit()`

Programm *Tooledit* laden:

```
AUX_PRGM.load_tooledit(<TOOLEFILE_PATH>)
```

`load_calibration()`

Programm zur *Kalibrierung* laden:

```
AUX_PRGM.load_calibration()
```

`keyboard_onboard()`

Laden der *onboard/Matchbox keyboard*

```
AUX_PRGM.keyboard_onboard(<ARGS>)
```

Anwendung

- **Modul `Aux_program_loader` importieren** +
  Fügen Sie diesen Python-Code zu Ihrem Importabschnitt hinzu:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.aux_program_loader import Aux_program_loader
```

- **Instantiierung des Moduls "Aux_program_loader " ** +
  Fügen Sie diesen Python-Code in Ihren "instantiate"-Abschnitt ein:

```
############################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
############################################

AUX_PRGM = Aux_program_loader()
```

12.9.10. `Keylookup`

Das Modul **`Keylookup`** wird verwendet, um **Tastendrücke zur Steuerung von Verhaltensweisen** zu ermöglichen wie z.B. Joggen.

Sie wird in der Handler-Datei verwendet, um die Erstellung von **Tastenbindungen** wie z. B. Tastatur-Jogging usw. zu erleichtern.

Anwendung

Modul `Keylookup` importieren

Um diese Module zu importieren, fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.keybindings import Keylookup
```

Instanziieren des Moduls `Keylookup`

Um das `Keylookup`-Modul* zu instanziieren, damit Sie es verwenden können, fügen Sie diesen Python-Code zu Ihrem Instanziierungsabschnitt hinzu:

```
############################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
############################################
```

```
KEYBIND = Keylookup()
```

NOTE

Hinzufügen von Tastenbelegungen

Keylookup erfordert Code unter der Funktion *processed_key_event*, um *KEYBIND.call()* aufzurufen.
Die meisten Handlerdateien verfügen bereits über diesen Code.

In der Handler-Datei, unter der *initialisierten Funktion* verwenden Sie diese allgemeine Syntax, um **Tastenbindungen** zu erstellen:

```
KEYBIND.add_call("DEFINED_KEY","FUNCTION TO CALL", USER DATA)
```

Hier fügen wir eine Tastenbindung für `F10`, `F11` und `F12` hinzu:

```
##########################################
# Spezielle Funktionen, die von QtVCP aufgerufen werden
##########################################

# zu diesem Zeitpunkt:
# sind die Widgets instanziiert.
# die HAL-Pins sind gebaut, aber HAL ist nicht bereit
def initialized__(self):
    KEYBIND.add_call('Key_F10','on_keycall_F10',None)
    KEYBIND.add_call('Key_F11','on_keycall_override',10)
    KEYBIND.add_call('Key_F12','on_keycall_override',20)
```

Und dann müssen wir **die Funktionen hinzufügen, die aufgerufen werden**.
Fügen Sie in der Handler-Datei unter dem Abschnitt `KEY BINDING CALLS` Folgendes hinzu:

```
####################
# KEY BINDING CALLS #
####################

def on_keycall_F12(self,event,state,shift,cntrl,value):
    if state:
        print('F12 pressed')

def on_keycall_override(self,event,state,shift,cntrl,value):
    if state:
        print('value = {}'.format(value))
```

Tastenbelegungen

Hier finden Sie eine Liste mit anerkannten Schlüsselwörtern. Verwenden Sie den zitierten Text.
Buchstabenschlüssel verwenden *Key_* mit dem hinzugefügten Groß- oder Kleinbuchstaben.
z. B. *Key_a* und *Key_A*.

```
keys = {
```

```
    Qt.Key_Escape: "Key_Escape",
    Qt.Key_Tab: "Key_Tab",
    Qt.Key_Backtab: "Key_Backtab",
    Qt.Key_Backspace: "Key_Backspace",
    Qt.Key_Return: "Key_Return",
    Qt.Key_Enter: "Key_Enter",
    Qt.Key_Insert: "Key_Insert",
    Qt.Key_Delete: "Key_Delete",
    Qt.Key_Pause: "Key_Pause",
    Qt.Key_Print: "Key_Print",
    Qt.Key_SysReq: "Key_SysReq",
    Qt.Key_Clear: "Key_Clear",
    Qt.Key_Home: "Key_Home",
    Qt.Key_End: "Key_End",
    Qt.Key_Left: "Key_Left",
    Qt.Key_Up: "Key_Up",
    Qt.Key_Right: "Key_Right",
    Qt.Key_Down: "Key_Down",
    Qt.Key_PageUp: "Key_PageUp",
    Qt.Key_PageDown: "Key_PageDown",
    Qt.Key_Shift: "Key_Shift",
    Qt.Key_Control: "Key_Control",
    Qt.Key_Meta: "Key_Meta",
    # Qt.Key_Alt: "Key_Alt",
    Qt.Key_AltGr: "Key_AltGr",
    Qt.Key_CapsLock: "Key_CapsLock",
    Qt.Key_NumLock: "Key_NumLock",
    Qt.Key_ScrollLock: "Key_ScrollLock",
    Qt.Key_F1: "Key_F1",
    Qt.Key_F2: "Key_F2",
    Qt.Key_F3: "Key_F3",
    Qt.Key_F4: "Key_F4",
    Qt.Key_F5: "Key_F5",
    Qt.Key_F6: "Key_F6",
    Qt.Key_F7: "Key_F7",
    Qt.Key_F8: "Key_F8",
    Qt.Key_F9: "Key_F9",
    Qt.Key_F10: "Key_F10",
    Qt.Key_F11: "Key_F11",
    Qt.Key_F12: "Key_F12",
    Qt.Key_F13: "Key_F13",
    Qt.Key_F14: "Key_F14",
    Qt.Key_F15: "Key_F15",
    Qt.Key_F16: "Key_F16",
    Qt.Key_F17: "Key_F17",
    Qt.Key_F18: "Key_F18",
    Qt.Key_F19: "Key_F19",
    Qt.Key_F20: "Key_F20",
    Qt.Key_F21: "Key_F21",
    Qt.Key_F22: "Key_F22",
    Qt.Key_F23: "Key_F23",
    Qt.Key_F24: "Key_F24",
    Qt.Key_F25: "Key_F25",
    Qt.Key_F26: "Key_F26",
    Qt.Key_F27: "Key_F27",
    Qt.Key_F28: "Key_F28",
```

```
    Qt.Key_F29: "Key_F29",
    Qt.Key_F30: "Key_F30",
    Qt.Key_F31: "Key_F31",
    Qt.Key_F32: "Key_F32",
    Qt.Key_F33: "Key_F33",
    Qt.Key_F34: "Key_F34",
    Qt.Key_F35: "Key_F35",
    Qt.Key_Super_L: "Key_Super_L",
    Qt.Key_Super_R: "Key_Super_R",
    Qt.Key_Menu: "Key_Menu",
    Qt.Key_Hyper_L: "Key_HYPER_L",
    Qt.Key_Hyper_R: "Key_Hyper_R",
    Qt.Key_Help: "Key_Help",
    Qt.Key_Direction_L: "Key_Direction_L",
    Qt.Key_Direction_R: "Key_Direction_R",
    Qt.Key_Space: "Key_Space",
    Qt.Key_Any: "Key_Any",
    Qt.Key_Exclam: "Key_Exclam",
    Qt.Key_QuoteDbl: "Key_QuoteDdl",
    Qt.Key_NumberSign: "Key_NumberSign",
    Qt.Key_Dollar: "Key_Dollar",
    Qt.Key_Percent: "Key_Percent",
    Qt.Key_Ampersand: "Key_Ampersand",
    Qt.Key_Apostrophe: "Key_Apostrophe",
    Qt.Key_ParenLeft: "Key_ParenLeft",
    Qt.Key_ParenRight: "Key_ParenRight",
    Qt.Key_Asterisk: "Key_Asterisk",
    Qt.Key_Plus: "Key_Plus",
    Qt.Key_Comma: "Key_Comma",
    Qt.Key_Minus: "Key_Minus",
    Qt.Key_Period: "Key_Period",
    Qt.Key_Slash: "Key_Slash",
    Qt.Key_0: "Key_0",
    Qt.Key_1: "Key_1",
    Qt.Key_2: "Key_2",
    Qt.Key_3: "Key_3",
    Qt.Key_4: "Key_4",
    Qt.Key_5: "Key_5",
    Qt.Key_6: "Key_6",
    Qt.Key_7: "Key_7",
    Qt.Key_8: "Key_8",
    Qt.Key_9: "Key_9",
    Qt.Key_Colon: "Key_Colon",
    Qt.Key_Semicolon: "Key_Semicolon",
    Qt.Key_Less: "Key_Less",
    Qt.Key_Equal: "Key_Equal",
    Qt.Key_Greater: "Key_Greater",
    Qt.Key_Question: "Key_Question",
    Qt.Key_At: "Key_At",
    Qt.Key_BracketLeft: "Key_BracketLeft",
    Qt.Key_Backslash: "Key_Backslash",
    Qt.Key_BracketRight: "Key_BracketRight",
    Qt.Key_AsciiCircum: "Key_AsciiCircum",
    Qt.Key_Underscore: "Key_Underscore",
    Qt.Key_QuoteLeft: "Key_QuoteLeft",
    Qt.Key_BraceLeft: "Key_BraceLeft",
```

```
    Qt.Key_Bar: "Key_Bar",
    Qt.Key_BraceRight: "Key_BraceRight",
    Qt.Key_AsciiTilde: "Key_AsciiTilde",

}
```

12.9.11. Messages

Das Modul **Messages** dient zur **Anzeige von Pop-up-Dialogmeldungen auf dem Bildschirm**.

Diese Nachrichten sind:

- *definiert in der INI-Datei unter der Überschrift `[DISPLAY]`*, und
- *gesteuert durch HAL-Pins*.

Verwenden Sie diesen Stil, wenn für jede Dialognachricht unabhängige HAL-Pins benötigt werden.

Eigenschaften

BOLDTEXT

Ist im Allgemeinen ein Titel.

TEXT

Text unter dem Titel und in der Regel länger.

DETAIL

Text ausgeblendet, sofern nicht angeklickt.

PINNAME

Basisname der HAL-Pin(s).

TYPE

Gibt an, ob es sich um handelt um (kann gleichzeitig eine Diaglo- und eine Status-Option haben):

- status* - angezeigt in der *Statusleiste und im Benachrichtigungsdialog*.
  Erfordert keinen Benutzereingriff.
- **nonedialog** - legt ausdrücklich fest, keinen Dialog zu zeigen.
- okdialog* - *die den Benutzer auffordert, auf OK zu klicken, um den Dialog zu schließen.*
  OK-Meldungen haben *zwei HAL-Pins*:
  - Ein HAL-Pin, um den Dialog zu starten, und
  - einen, um anzuzeigen, dass auf eine Antwort gewartet wird. **yesnodialog** - zwingt denBenutzer, Ja- oder Nein-Schaltflächen zum Schließen des Dialogs auszuwählen_
    Ja/Nein-Nachrichten haben *drei HAL Pins*:
  - Einen, um den Dialog anzuzeigen,
  - Einen für das Warten, und

    - eine für die Antwort. **message'okcanceldialog** - *zwingt den Benutzer, OK oder Abbrechen zum Schließen des Dialogs auszuwählen*
      Ja/Nein-Nachrichten haben *drei HAL Pins*:
    - Einen, um den Dialog anzuzeigen,
    - Einen für das Warten, und
    - eine für die Antwort.
- ***closepromptdialog*** - *verlangt eine Auswahl des Anwenders*

Standardmäßig werden `STATUS`-Meldungen für `focus_overlay` und den Warnsound gesendet, sobald der Dialog angezeigt wird.
So können dem Bildschirm „Fokus"-Abdunkelung/Unschärfe und Töne zu Warnungen hinzugefügt werden.

HAL-Pins

Die HAL Pin Namen würden dieses Muster verwenden:

<SCREEN BASENAME>.<PINNAME>
: invoking sint pin

<SCREEN BASENAME>.<PINNAME>-waiting
: *Waiting for the user's response* output bool pin

<SCREEN BASENAME>.<PINNAME>-response
: *The user response* output bool pin

<SCREEN BASENAME>.<PINNAME>-response-s32
: *The user response* output sint pin

Beispiele

Hier sind Beispiele für Codeblöcke zur Definition von INI-Nachrichten, die unter der Überschrift '[DISPLAY]' zu finden sind:

- Statusleiste und Desktop-Benachrichtigungs-Pop-up-Meldung:

```
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a statusbar test
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
```

- Pop-up-Dialog mit einer Ja/Nein-Frage:

```
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_DETAILS = Y/N DETAILS
```

```
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yndialogtest
```

- Pop-up-Dialog, der eine OK-Antwort verlangt + Statusleiste und Desktop-Benachrichtigung:

```
[DISPLAY]
MESSAGE_BOLDTEXT = Dies ist der kurze Text
MESSAGE_TEXT = Dies ist der längere Text des Tests der beiden Typen. Er kann länger
sein als der Text der Statusleiste
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog status
MESSAGE_PINNAME = bothtest
```

Das Widget `ScreenOptions` kann das Nachrichtensystem automatisch einrichten.

12.9.12. multimessages

Das Modul **`Messages`** dient zur **Anzeige von Pop-up-Dialogmeldungen auf dem Bildschirm**.

Diese Nachrichten sind:

- *definiert in der INI-Datei unter der Überschrift `[DISPLAY]`*, und
- *controlled by one sint HAL pin per defined id.*
- *each message is called by a corresponding number on the sint pin.*

Verwenden Sie diesen Stil von Benutzerinformationen beispielsweise, wenn ein VFD Fehlermeldungen als Zahlen kodiert sendet.
Er verwendet gemeinsame Invoke/Response/Wait-HAL-Pins für alle (pro ID-Name) Multimessage-Dialoge. Die HAL-Pin-Namen würden diese Muster verwenden:

<SCREEN BASENAME>.<ID NAME>

invoking sint pin

<SCREEN BASENAME>.<ID NAME>-waiting

Waiting for the user's response output bool pin

<SCREEN BASENAME>.<ID NAME>-response

The user response output bool pin

<SCREEN BASENAME>.<ID NAME>-response-s32

The user response output sint pin

Eigenschaften

`TITLE`

Dies ist der Titel des Dialogfensters.

TEXT

Text unter dem Titel und in der Regel länger.

DETAIL

Text ausgeblendet, sofern nicht angeklickt.

TYPE

Gibt den Typ der Nachricht an, den der Benutzer sieht (Dialog- und Statusoptionen können kombiniert werden):

- status* - angezeigt in der *Statusleiste und im Benachrichtigungsdialog*.
  Erfordert keinen Benutzereingriff.
- **nonedialog** - legt ausdrücklich fest, keinen Dialog zu zeigen.
  okdialog - *zwingt den Benutzer, auf OK zu klicken, um den Dialog zu schließen.*
  OK-Meldungen haben *zwei HAL-Pins*:
  - Ein HAL-Pin, um den Dialog zu starten, und
  - einen, um anzuzeigen, dass auf eine Antwort gewartet wird.
- yesnodialog* (Ja / Nein-Nachricht) - *zwingt den Benutzer zu einem Ja- oder Nein zum Schließen des Dialogs*
  Ja/Nein-Nachrichten haben *drei HAL Pins*:
  - Einen, um den Dialog anzuzeigen,
  - Einen für das Warten, und
  - eine für die Antwort.

Standardmäßig werden `STATUS`-Meldungen für `focus_overlay` und den Warnsound gesendet, sobald der Dialog angezeigt wird.
So können dem Bildschirm „Fokus“-Abdunkelung/Unschärfe und Töne zu Warnungen hinzugefügt werden.

Beispiele

Hier sind Beispiele für Codeblöcke zur Definition von INI-Nachrichten, die unter der Überschrift ‘[DISPLAY]’ zu finden sind:

```
[DISPLAY]
MULTIMESSAGE_ID = VFD

MULTIMESSAGE_VFD_NUMBER = 1
MULTIMESSAGE_VFD_TYPE = okdialog status
MULTIMESSAGE_VFD_TITLE = VFD Fehler: 1
MULTIMESSAGE_VFD_TEXT = Die ist der längere Text FÜR MESSAGE NUMMER 1
MULTIMESSAGE_VFD_DETAILS = DETAILS für VFD Fehler 1
MULTIMESSAGE_VFD_ICON = WARNING

MULTIMESSAGE_VFD_NUMBER = 2
MULTIMESSAGE_VFD_TYPE = nonedialog status
MULTIMESSAGE_VFD_TITLE = VFD Fehler: 2
```

```
MULTIMESSAGE_VFD_TEXT = Die sist der längere Text FÜR MESSAGE NUMMER 2
MULTIMESSAGE_VFD_DETAILS = DETAILS für VFD Fehler 2
MULTIMESSAGE_VFD_ICON = INFO
```

12.9.13. Notify

Das Modul **Notify** wird verwendet, um **Nachrichten zu versenden, die in den Desktop integriert** sind.

Es verwendet die pynotify Bibliothek.

Ubuntu/Mint folgt nicht dem Standard, so dass man nicht einstellen kann, wie lange die Meldung angezeigt wird.
Ich schlage vor, dies mit dem Paket notify-osd zu beheben, das unter dieses PPA verfügbar ist (DISCONTINUED aufgrund der Umstellung von Ubuntu auf Gnome).

Notify *erhält eine Liste aller Alarmmeldungen seit dem Start* in **self.alarmpage**.
Wenn Sie im Notify-Popup auf *'Show all messages'* klicken, werden sie auf dem Terminal ausgegeben.

Das Widget ScreenOptions kann das Benachrichtigungssystem automatisch einrichten.

Typischerweise werden STATUS *messages* verwendet, um Benachrichtigungen zu senden.

Eigenschaften

Sie können Folgendes festlegen:

title

Titeltext der Benachrichtigung.

message

Inhalt der Benachrichtigungsnachricht.

icon

Symbol für eine Benachrichtigung.

timeout

Wie lange die Nachricht angezeigt wird.

12.9.14. Preferences

Das Modul **Preferences** ermöglicht das **Laden und dauerhafte Speichern von Einstellungsdaten von/auf Speichermedien**.

Das Widget ScreenOptions kann das Einstellungssystem automatisch einrichten.

QtVCP sucht zuerst nach dem ScreenOptions-Widget und ruft, falls gefunden, **_pref_init()** auf.
Dies *erzeugt das Einstellungsobjekt* und gibt es an QtVCP zurück, um es an alle Widgets zu übergeben und es zu den Attributen des Fensterobjekts hinzuzufügen.

In diesem Fall wäre das Einstellungsobjekt von der `initialized_`-Methode der Handler-Datei als **`self.w.PREFS_`** zugänglich.

Außerdem können alle Widgets bei der Initialisierung Zugriff auf eine bestimmte Einstellungsdatei haben.

Das Widget `ScreenOptions` kann die Einstellungsdatei automatisch einrichten.

12.9.15. `Player`

Dieses Modul **ermöglicht das Abspielen von Sounds mit Gstreamer, Beep und Espeak**.

Es kann:

- **Abspielen von Sound-/Musikdateien** mit *Gstreamer* (nicht blockierend),
- **Sounds abspielen** unter Verwendung der `beep`-Bibliothek (blockiert derzeit beim Piepsen),
- **Sprachwörter** unter Verwendung der `espeak`-Bibliothek (keine Blockierung beim Sprechen).

Es gibt *Standard-Warntöne*, die Mint oder FreeDesktop-Standardsounds verwenden.

Sie können beliebige Sounds oder sogar Songs abspielen, indem Sie den Pfad angeben.

`STATUS` hat *Nachrichten zur Steuerung des `Player`-Moduls*.

Das Widget `ScreenOptions` kann automatisch das Audiosystem einrichten.

Töne (engl. sounds)

Alarmsignale

Es gibt Standard-**Warnungen** zur Auswahl:

- `ERROR`
- `READY`
- `ATTENTION`
- `RING`
- `DONE`
- `LOGIN`
- `LOGOUT`

Pieptöne

Es gibt drei **Pieptöne**:

- `BEEP_RING`
- `BEEP_START`
- `BEEP`

Anwendung

- **Import `Player` module**
  Fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.audio_player import Player
```

- **Instantiierung des Moduls `Player`**
  Fügen Sie diesen Python-Code zu Ihrem instanziierten Abschnitt hinzu:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

SOUND = Player()
SOUND._register_messages()
```

  Die Funktion **`_register_messages()`** verbindet den Audioplayer mit der `STATUS`-Bibliothek, so dass Klänge mit dem `STATUS`-Meldungssystem abgespielt werden können.

Beispiel

Um Töne mit `STATUS`-Meldungen abzuspielen, verwenden Sie diese allgemeine Syntax:

```
STATUS.emit('play-alert','LOGOUT')
STATUS.emit('play-alert','BEEP')
STATUS.emit('play-alert','SPEAK This is a test screen for Q t V C P')
STATUS.emit('play-sound', 'PATH TO SOUND')
```

12.9.16. Virtuelle Tastatur

Diese Bibliothek ermöglicht es Ihnen, **mit `STATUS`-Nachrichten eine virtuelle Tastatur zu starten**.

Es verwendet `Onboard` oder `Matchbox` Bibliotheken für die Tastatur.

12.9.17. Aktionen in der Symbolleiste

Diese Bibliothek liefert **vorgefertigte Untermenüs und Aktionen für Symbolleistenmenüs und Symbolleistenschaltflächen**.

Werkzeug-Buttons, Menüs und Symbolleistenmenüs sind:

- *in Qt Designer erstellt*, und
- *zugewiesene Aktionen/Untermenüs in der Handler-Datei.*

Aktionen

`estop` (engl. für Notaus)

`power`

`load`

`reload`

`gcode_properties`

`run`

`pause`

`abort`

`block_delete` (engl. für Block löschen)

`optional_stop`

`touchoffworkplace`

`touchofffixture`

`runfromline` (Ausführen von gegebener Zeile)

`load_calibration`

`load_halmeter`

`load_halshow`

`load_status`

`load_halscope`

`about`

`zoom_in`

`zoom_out`

`view_x`

`view_y`

`view_y2`

`view_z`

`view_z2`

`view_p`

`view_clear`

`show_offsets`

`quit`

`system_shutdown`

`tooloffsetdialog`

`originoffsetdialog`

`calculatordialog`

`alphamode`

`inhibit_selection`

`show_dimensions`

Schaltet die Anzeige der Dimensionen um.

Untermenüs

recent_submenu

home_submenu

unhome_submenu

zero_systems_submenu

grid_size_submenu

Menü zum Einstellen der Größe des Grafikrasters

Anwendung

Hier ist der typische Code, der zu den entsprechenden Abschnitten der *Handler-Datei* hinzuzufügen ist:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.toolbar_actions import ToolBarActions

########################################
**** Bibliotheken instanziieren Abschnitt **** #
########################################

TOOLBAR = ToolBarActions()
```

Beispiele

- Zuweisung von Werkzeugaktionen zu Buttons der Symbolleiste

```
########################################
# Spezielle Funktionen, die von QtVCP aufgerufen werden
########################################

# An diesem Punkt:
# * sind die Widgets instanziiert,
# * die HAL-Pins sind gebaut, aber HAL ist nicht bereit.
def initialized__(self):
    TOOLBAR.configure_submenu(self.w.menuHoming, 'home_submenu')
    TOOLBAR.configure_action(self.w.actionEstop, 'estop')
    TOOLBAR.configure_action(self.w.actionQuit, 'quit', lambda d:self.w.close())
    TOOLBAR.configure_action(self.w.actionEdit, 'edit', self.edit)
    # Add a custom function
    TOOLBAR.configure_action(self.w.actionMyFunction, 'my_Function', self.my_function)
```

- Hinzufügen einer benutzerdefinierten Symbolleistenfunktion:

```
####################
# GENERAL FUNCTIONS #
####################

def my_function(self, widget, state):
```

```
    print('My function State = ()'.format(state))
```

12.9.18. Qt Vismach Maschinengrafik-Bibliothek

Qt_vismach ist ein *Satz von Python-Funktionen*, der dazu verwendet werden kann, **Modelle von Maschinen zu erstellen und zu animieren**.

Vismach:

- *zeigt das Modell* in einem **3D-Ansichtsfenster**
- *animiert die Modellteile* während sich die Werte der zugehörigen HAL-Pins ändern.

Dies ist die *Qt basierte Version* der Bibliothek, es gibt auch eine tkinter Version, die in LinuxCNC verfügbar ist.

Die Qt-Version *ermöglicht die Einbettung der Simulation in andere Bildschirme*.

Integrierte Beispiele

In QtVCP sind *Beispielpanels* enthalten für:

- eine 3-Achsen-XYZ-Fräse,
- eine 5-Achsen-Portalfräse,
- eine 3-Achsen-Fräse mit einer A-Achse/Spindel und
- eine Scara-Fräse.

Die meisten dieser Beispiele, wenn sie nach einer laufenden LinuxCNC-Konfiguration geladen werden (einschließlich nicht-QtVCP-basierter Bildschirme), reagieren auf Maschinenbewegungen.
Einige erfordern, dass HAL-Pins für die Bewegung angeschlossen werden.

Von einem Terminal aus (wählen Sie eines aus):

```
qtvcp vismach_mill_xyz
qtvcp vismach_scara
qtvcp vismach_millturn
qtvcp vismach_5axis_gantry
```

Primitives-Bibliothek

Stellt die **grundlegenden Bausteine einer simulierten Maschine** bereit.

Collection

Eine Sammlung ist ein **Gegenstand einzelner Maschinenteile**.

Diese enthält eine **hierarchische Liste** von primitiven Formen oder *STL-Objekten*, auf die Operationen angewendet werden können.

Translate

Dieses Objekt führt eine **OpenGL-Translation** für ein Sammelobjekt aus.

Unter Translation versteht man das *geradlinige Verschieben* eines Objekts an eine andere Position auf dem Bildschirm.

Scale

Dieses Objekt führt eine **OpenGL-Skalierungsfunktion** *für ein Sammelobjekt* aus.

HalTranslate

Dieses Objekt führt eine **OpenGL-Verschiebung** (engl. translation) *für ein Sammelobjekt* durch, **versetzt um den Wert eines HAL-Pins**.

Unter Translation versteht man das Verschieben eines Objekts in gerader Linie an eine andere Position auf dem Bildschirm.

Sie können entweder:

- *einen Pin von einer Komponente lesen, die dem Vismach-Objekt gehört*, oder
- *direktes Lesen eines HAL-Systempins*, wenn das Komponentenargument auf `None` gesetzt ist.

Rotate

Dieses Objekt führt eine **OpenGL-Rotationsberechnung** *für ein Sammelobjekt* durch.

HalRotate

Dieses Objekt führt die Berechnung für eine **OpenGL-Drehung*eines Sammelobjekts* durch, *versetzt um den HAL-Pin-Wert**.

Sie können entweder:

- *read einen Pin von einer Komponente, die dem vismach* Objekt gehört, oder
- *direktes Lesen eines HAL-Systempins*, wenn das Komponentenargument auf `None` gesetzt ist.

HalToolCylinder

Dieses Objekt erstellt eine *CylinderZ object*, die **Größe und Länge basierend auf der geladenen Werkzeugdefinition** (aus der Werkzeugtabelle) ändert

Es liest die *HAL-Pins* `halui.tool.diameter` und `motion.tooloffset.z`.

Beispiel aus dem mill_xyz-Beispiel:

```
toolshape = CylinderZ(0)
toolshape = Color([1, .5, .5, .5], [toolshape])
tool = Collection([
    Translate([HalTranslate([tooltip], None, "motion.tooloffset.z", 0, 0, -
MODEL_SCALING)], 0, 0, 0),
    HalToolCylinder(toolshape)
])
```

`Track` (engl. für Spur)

Bewege und drehe ein Objekt, um von einer `capture()`-Koordinate aus zu zeigen System zu einem anderen.

Basisobjekt zur *Aufnahme von Koordinaten für primitive Formen*.

`CylinderX`, `CylinderY`, `CylinderZ`

*Konstruieren Sie einen Zylinder auf der X-, Y- oder Z-Achse, indem Sie den *Endpunkt* (X, Y oder Z) und *Radien* Koordinaten.

`Sphere` (engl. für Kugel)

Baue eine Kugel aus den Koordinaten *center* und *radius*.

`TriangleXY`, `TriangleXZ`, `TriangleYZ`

Baue ein Dreieck in der *angegebenen Ebene* durch Angabe der Z-Koordinaten der Eckpunkte_ für jede Seite.

`ArcX`

Bogen erstellen durch Angabe

`Box`

Erstellen Sie eine Box, die durch den coordinates_ _6-Scheitelpunkt angegeben wird.

`BoxCentered`

Erstellen einer Box zentriert auf Ursprung durch Angabe der *Breite/Tiefe in X und Y,* und deren *Höhe in Z*.

`BoxCenteredXY`

Erstelle eine Box, die in X und Y zentriert ist und von Z=0 ausgeht, indem man die *Breite in X und Y* angibt und nach oben oder unten bis zur angegebenen *Höhe in Z* läuft.

`Capture`

Erfassen der aktuellen Transformationsmatrix einer Sammlung.

NOTE	Dies *transformiert vom aktuellen Koordinatensystem in das System des Ansichtsfensters*, NICHT in das Weltsystem.

`Hud`

Heads up display zeichnet ein *semi-transparentes Textfeld*.

Verwendung:

- `HUD.strs` für Dinge, die *ständig* aktualisiert werden müssen,
- `HUD.show("stuff")` für einmalige Dinge wie Fehlermeldungen.

`Color`

Wendet eine Farbe an auf die *Teile einer Collection*.

AsciiSTL, **AsciiOBJ**

Lädt eine STL- oder OBJ-Datendatei als *Vismach part*.

Anwendung

Importieren einer Simulation

So könnte man die XYZ_mill-Simulation in eine QtVCP-Panel- oder Screenhandler-Datei importieren.

```
############################
# **** IMPORT SECTION **** #
############################

import mill_xyz as MILL
```

Instanziierung und Verwendung des Simulations-Widgets

Instanziieren Sie das Simulations-Widget und fügen Sie es dem Hauptlayout des Bildschirms hinzu:

```
########################################
# Spezielle Funktionen, die von QtVCP aufgerufen werden
########################################

# Zu diesem Zeitpunkt:
#   * sind die Widgets instanziiert.
#   * die HAL-Pins sind gebaut, aber HAL ist nicht bereit
def initialized__(self):
    machine = MILL.Window()
    self.w.mainLayout.addWidget(maschine)
```

Mehr zum Thema

Weitere Informationen über die Erstellung einer benutzerdefinierten Maschinensimulation finden Sie im Kapitel zu Qt Vismach.

12.10. QtVismach

Vismach ist ein Satz von **Python-Funktionen, mit denen man Maschinenmodelle erstellen und Modelle von Maschinen** zu erstellen und zu animieren.

In diesem Kapitel geht es um die in Qt eingebettete Version von Vismach, siehe auch: https://sa-cnc.com/linuxcnc-vismach/ .

12.10.1. Einführung

Vismach zeigt das Modell in einem **3D-Ansichtsfenster** (engl. viewport) an und die **Modellteile werden animiert, wenn sich die Werte der zugehörigen HAL-Pins ändern**.

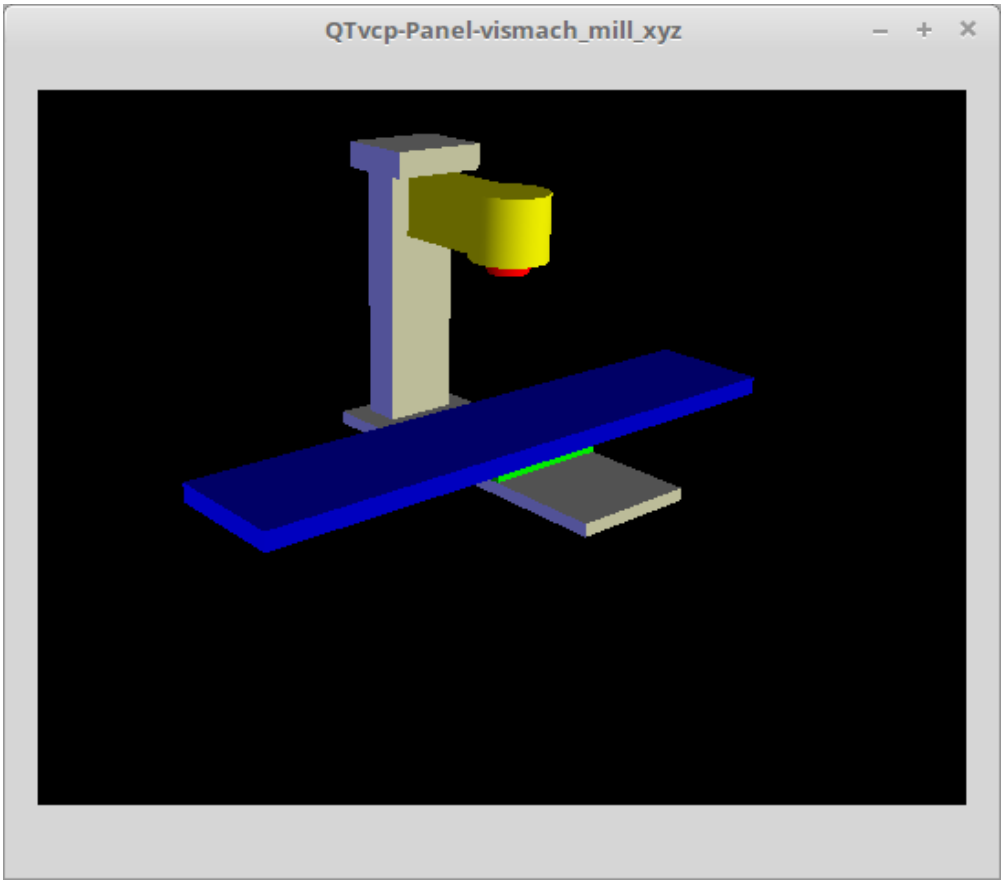


Figure 330. QtVismach 3D Ansicht

Das Vismach 3D Ansichtsfenster (engl. viewport view) kann wie folgt manipuliert werden:

- **Zoom** durch *Scrollrad*
- **schwenk** durch *Ziehen mit der mittleren Taste*
- **drehen** durch *Ziehen mit der rechten Maustaste*
- **kippen** (engl. tilt) durch *linke Maustaste ziehen*

Ein **Vismach-Modell** hat die Form eines *Python-Skripts* und kann die Standard-Python-Syntax verwenden.

Das bedeutet, dass es mehr als eine Möglichkeit gibt, das Skript zu gestalten, aber in den Beispielen in diesem Dokument wird die einfachste und grundlegendste davon verwendet.

Die grundlegende Reihenfolge bei der Erstellung des Vismach-Modells ist:

1. Erstellen der Teile
2. Definieren, wie sie sich bewegen
3. In Bewegungsgruppen zusammenstellen

12.10.2. Hierarchie des Maschinendesigns

Das Modell folgt einer **logischen Baumstruktur**.

Stellen Sie sich einen Baum vor, mit Wurzel/Stamm, Ästen und kleineren Zweigen. Wenn Sie den größeren Ast bewegen, dann bewegen sich die kleineren Äste mit, aber wenn Sie den kleineren Ast bewegen, so bewegen sich die größeren dennoch nicht.

Das Maschinendesign folgt diesem konzeptionellen Design.

Nehmen wir als Beispiel die Fräse, die im obigen Bild des 3D-Ansichtsfensters zu sehen ist:

- Wenn Sie X bewegen, kann sie sich von selbst bewegen,
- aber wenn Sie die Baugruppe Y verschieben, wird auch die Baugruppe X verschoben, da sie mit der Baugruppe Y verbunden ist.

Für diese Maschine sieht der Baum so aus:

```
model
  |
  |---frame
  |     |
  |     |---base
  |     |
  |     |---column
  |     |
  |     |---top
  |
  |---yassembly
  |       |
  |       |---xassembly
  |       |       |
  |       |       |---xbase
  |       |       |
  |       |       |---work
  |       |
  |       |---ybase
  |
  |---zassembly
           |
           |---zframe
           |     |
           |     |---zbody
           |     |
           |     |---spindle
           |
           |---toolassembly
                     |
                     |---cat30
                     |
                     |---tool
                           |
                           |---tooltip
                           |
                           |---(tool cylinder function)
```

Wie Sie sehen, muss das *niedrigste Teil zuerst existieren, bevor es mit anderen zu einer Baugruppe*

zusammengefasst werden kann. Sie *bauen* also vom niedrigsten Punkt im Baum aufwärts und fügen sie zusammen.

Dasselbe gilt für jede Art von Maschinenkonstruktion: Schauen Sie sich das Beispiel des Maschinenarms an, und Sie werden sehen, dass er mit der Spitze beginnt und sich zum größeren Teil des Arms hinzugesellt, um sich schließlich mit der Basis zu vereinen.

12.10.3. Starten des Skripts

Zum Testen ist es nützlich, die Shebang-Zeile `#!/usr/bin/env python3` einzufügen, damit die Datei direkt von der Kommandozeile ausgeführt werden kann.

Zunächst sind *die erforderlichen Bibliotheken zu importieren*.

```
#!/usr/bin/env python3

import hal
import math
import sys

from qtvcp.lib.qt_vismach.qt_vismach import *
```

12.10.4. HAL-Pins.

Ursprünglich erforderte die Vismach-Bibliothek die Erstellung einer Komponente und den Anschluss von HAL-Pins zur Steuerung der Simulation.

`qt_vismach` kann die HAL-Systempins direkt lesen oder, wenn Sie es wünschen, separate HAL-Pins verwenden, die Sie in einer HAL-Komponente definieren müssen:

```
c = hal.component("samplegui")
c.newpin("joint0", hal.Type.REAL, hal.Dir.IN)
c.newpin("joint1", hal.Type.REAL, hal.Dir.IN)
c.ready()
```

Sie können zwischen den beiden Optionen in den Funktionen wählen, die diese Einträge verwenden:

`hal_comp`

Die *HAL Komponenten Objekt* oder None.
In QtVCP, wenn Sie *System-Pins* direkt lesen, wird das Komponentenargument auf `None` gesetzt.

`hal_pin`

The *name of the BOOL HAL IN pin* that will change the color.
If hal_comp is *None* then this must be the full name of a system pin otherwise this is the pin name excluding the component name.

12.10.5. Erstellen von Teilen

Importieren von STL- oder OBJ-Dateien

Das ist wahrscheinlich am einfachsten:

- Herstellung einer Geometrie in einem CAD-Paket_
- _Import in das Modellskript unter Verwendung der Funktionen "ASCIISTL()" oder "ASCIIOBJ()".

Beide Funktionen können eines von zwei benannten Argumenten annehmen, entweder einen *Dateinamen* oder *Daten*:

```
part = AsciiSTL(filename="path/to/file.stl")
part = AsciiSTL(data="solid part1 facet normal ...")
part = AsciiOBJ(filename="path/to/file.obj")
part = AsciiOBJ(data="v 0.123 0.234 0.345 1.0 ...")
```

- STL-Modellteile werden dem Vismach-Raum an *denselben Stellen hinzugefügt, an denen sie im STL- oder OBJ-Raum erstellt wurden*, d. h. idealerweise mit einem Rotationspunkt an ihrem Ursprung.

NOTE Es ist viel einfacher, sich während des Bauens zu bewegen, wenn der Ursprung des Modells an einem Drehpunkt liegt.

Aufbau aus geometrischen Primitiven

Alternativ können Teile auch *im Modellskript aus einer Reihe von Formprimitiven* erstellt werden.

`assembly = collction([part1,part2,part3])`

Collection ist ein allgemeiner Container von miteinander in Beziehung stehenden (endl. related)Teilen (engl. parts)

Viele Formen werden *am Ursprung* erstellt und müssen nach der Erstellung *an den gewünschten Ort* verschoben werden.

`cylinder = CylinderX(x1, r1, x2, r2)`

`cylinder = CylinderY(y1, r1, y2, r2)`

`cylinder = CylinderZ(z1, r1, z2, r2)`

Erzeugt einen *(optional verjüngten) Zylinder auf der gegebenen Achse* mit den gegebenen Radien an den gegebenen Punkten auf der Achse.

`sphere = Sphere(x, y, z, r)`

Erzeugt eine *Kugel mit Radius r bei (x,y,z).*

`triangle = TriangleXY(x1, y1, x2, y2, x3, y3, z1, z2)`

`triangle = TriangleXZ(x1, z1, x2, z2, x3, z3, y1, y2)`

`triangle = TriangleYZ(y1, z1, y2, z2, y3, z3, x1, x2)`

Erzeugt eine *dreieckige Platte zwischen Ebenen*, die durch die letzten beiden Werte parallel zur

angegebenen Ebene definiert ist und deren Eckpunkte durch die drei Koordinatenpaare gegeben sind.

`arc = ArcX(x1, x2, r1, r2, a1, a2)`

Erstellen Sie eine *Bogenform*.

`box = Box(x1, y1, z1, x2, y2, z2)`

Erzeugt ein *rechteckiges Prisma mit gegenüberliegenden Ecken* an den angegebenen Positionen und Kanten parallel zu den XYZ-Achsen.

`box = BoxCentered(xw, yw, zw)`

Erzeugt eine xw mal yw mal zw *Box, die auf den Ursprung zentriert ist*.

`box = BoxCenteredXY(xw, yw, z)`

Erstellt einen *Kastenboden auf der WY-Ebene* mit der Breite xw / yw und der Höhe z.

Zusammengesetzte Teile können durch Zusammenfügen dieser Primitive entweder zum Zeitpunkt der Erstellung oder später erstellt werden:

```
part1 = Collection([Sphere(100,100,100,50), CylinderX(100,40,150,30)])
part2 = Box(50,40,75,100,75,100)
part3 = Collection([part2, TriangleXY(10,10,20,10,15,20,100,101)])
part4 = Collection([part1, part2])
```

12.10.6. Bewegliche Teile des Modells

Möglicherweise müssen Teile im Vismach-Raum verschoben werden, um das Modell zusammenzusetzen. Der Ursprung bewegt sich nicht - Translate() und Rotate() verschieben die Collection, während Sie Teile hinzufügen, relativ zu einem stationären Ursprung.

Verschieben von Teilen des Modells

`part1 = Translate([part1], x, y, z)`

Verschiebe Teil1 um die angegebenen Abstände in x, y und z.

Rotation von Teilen des Modells

`part1 = Rotate([part1], theta, x, y, z)`

Drehen Sie das Teil um den Winkel theta [Grad] um eine Achse zwischen dem Ursprung und x, y, z.

12.10.7. Animieren von Teilen

Zur **Animation des Modells durch die Werte der HAL-Pins** gibt es vier Funktionen `HalTranslate`, `HalRotate`, `HalToolCylinder` und `HalToolTriangle`.

Damit sich Teile innerhalb einer Baugruppe bewegen können, müssen ihre HAL-Bewegungen definiert werden, bevor sie mit dem Befehl "Collection" zusammengebaut werden.

Die **Rotationsachse und der Translationsvektor bewegen sich mit dem Teil**:

- wie es vom Vismach-Skript während der Modellmontage verschoben wird, oder
- während es sich als Reaktion auf die HAL-Pins bewegt, während das Modell animiert wird.

HalTranslate

part = HalTranslate([part], hal_comp, hal_pin, xs, ys, zs)

part

Eine *Sammlung oder ein Teil*.
Sie kann zu einem früheren Zeitpunkt im Skript erstellt werden oder, falls gewünscht, an dieser Stelle, z. B.

```
`part1 = HalTranslate([Box(....)], ...)`. +
```

hal_comp

Die *HAL Komponente* ist das nächste Argument.
In QtVCP, wenn Sie *System-Pins* direkt lesen, wird das Komponentenargument auf `None` gesetzt.

hal_pin

Der *Name des HAL-Pins*, der die Bewegung animieren soll.
Dieser muss mit einem bestehenden HAL-Pin übereinstimmen, der die Gelenkposition beschreibt, wie z. B.:

```
"joint.2.pos-fb"
```

Andernfalls würde die Komponenteninstanz und der Pin-Name dieser Komponente angegeben werden.

xs, ys, zs

Die *X, Y, Z Skalen*.
Bei einer kartesischen Maschine, die im Maßstab 1:1 erstellt wurde, wäre dies normalerweise "1,0,0" für eine Bewegung in positiver X-Richtung.
Wenn die STL-Datei jedoch in cm und die Maschine in Zoll erstellt wurde, kann dies an dieser Stelle durch die Verwendung von 0,3937 (= 1 cm/1 Zoll = 1 cm /2,54 cm) als Maßstab festgelegt werden.

HalRotate

part = HalRotate([part], hal_comp, hal_pin, angle_scale, x, y, z)

Dieser Befehl funktioniert ähnlich wie `HalTranslate`, mit dem Unterschied, dass es normalerweise notwendig ist, das Teil zuerst zum Ursprung zu bewegen, um die Achse zu definieren.

x, y, z

Definiert die *Drehachse* vom Ursprung, dem Koordinatenpunkt (x,y,z).
Wenn das Teil vom Ursprung zurück an seine richtige Position bewegt wird, kann die Drehachse

als im Teil "eingebettet" betrachtet werden.

angle_scale

Drehwinkel werden in Grad angegeben. Für ein Drehgelenk mit einer Skalierung von 0-1 müssten Sie also eine Winkelskala von 360 verwenden.

HalToolCylinder

tool = HalToolCylinder()

Stellen Sie einen Zylinder her, der ein zylindrisches Fräswerkzeug darstellt, basierend auf dem Werkzeugtisch und dem aktuell belasteten Werkzeug.

```
tool = HalToolCylinder()
toolshape = Color([1, .5, .5, .5],[tool])

# oder kompakter:
toolshape = Color([1, .5, .5, .5], [HalToolCylinder()])
```

HalToolTriangle

tool = HalToolTriangle()

Erstellen Sie ein Dreieck, um ein dreieckiges Drehwerkzeug darzustellen, basierend auf dem Werkzeugtisch und dem aktuell geladenen Werkzeug.

```
tool = HalToolTriangle()
toolshape = Color([1, 1, 0, 1],[tool])

# oder kompakter:
toolshape = Color([1, 1, 0, 1],[HalToolTriangle()])
```

HAL einstellbare Grundelemente (engl. Primitives)

Alle Form-Grundelemente (engl. shape primitives) können HAL-Pin-Namen anstelle von Koordinaten verwenden.
Entweder indem das Komponentenobjekt als erste Variable hinzugefügt und der pinname-String für eine Koordinate eingesetzt wird, oder
indem der vollständige Komponenten-/Pinname-String für eine Koordinate eingesetzt wird.

Erzeugt ein *rechteckiges Prisma mit gegenüberliegenden Ecken* an den angegebenen Positionen und Kanten parallel zu den XYZ-Achsen.
Die Z Start-Koordinate wird kontrolliert durch den HAL-Pin *Zstart*.

```
box = Box(component, x1, y1, 'Zstart', x2, y2, z2)
box = Box(x1, y1, 'componentName.Zstart', x2, y2, z2)
```

12.10.8. Zusammenbau des Modells

Damit sich die Teile gemeinsam bewegen können, müssen sie mit dem Befehl `Collection()` zusammengefügt werden.

Es ist wichtig, **die Teile zusammenzufügen und ihre Bewegungen in der richtigen Reihenfolge zu definieren**.

Um beispielsweise eine Fräsmaschine mit beweglichem Kopf, einer rotierenden Spindel und einer animierten Zugstange zu erstellen, würden Sie dies tun:

- Erstellen Sie den Hauptteil des Kopfes.
- Erstellen Sie die Spindel im Ursprung.
- Definieren Sie die Drehung.
- Bewegen Sie den Kopf zur Spindel oder die Spindel zum Kopf.
- Erstellen Sie die Zugstange (engl. draw bar).
- Definieren Sie die Bewegung der Zugstange.
- Bauen Sie die drei Teile zu einer Kopfeinheit zusammen.
- Definieren Sie die Bewegung der Kopfeinheit.

In diesem Beispiel wird die Spindeldrehung durch die Drehung eines Satzes von Mitnehmern angezeigt:

```
#Drive dogs
dogs = Box(-6,-3,94,6,3,100)
dogs = Color([1,1,1,1],[dogs])
dogs = HalRotate([dogs],c,"spindle",360,0,0,1)
dogs = Translate([dogs],-1,49,0)

#Drawbar
draw = CylinderZ(120,3,125,3)
draw = Color([1,0,.5,1],[draw])
draw = Translate([draw],-1,49,0)
draw = HalTranslate([draw],c,"drawbar",0,0,1)

# head/spindle
head = AsciiSTL(filename="./head.stl")
head = Color([0.3,0.3,0.3,1],[head])
head = Translate([head],0,0,4)
head = Collection([head, tool, dogs, draw])
head = HalTranslate([head],c,"Z",0,0,0.1)

# base
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# mount head on it
base = Collection([head, base])
```

Schließlich muss eine **einzige Sammlung aller Maschinenteile, Böden und Arbeiten** (falls vorhanden) erstellt werden.

Für eine *serial machine* wird jedes neue Teil der Sammlung des vorherigen Teils hinzugefügt.

Bei einer *Parallelmaschine* kann es mehrere "Basis"-Teile geben.

So wird zum Beispiel in `scaragui.py` link3 zu link2, link2 zu link1 und link1 zu link0 hinzugefügt, so dass das endgültige Modell wie folgt erstellt wird:

```
model = Collection([link0, floor, table])
```

Ein VMC-Modell mit separaten Teilen, die sich auf dem Sockel bewegen, könnte hingegen haben

```
model = Collection([base, saddle, head, carousel])
```

12.10.9. Weitere Funktionen

Farbe

Legt die *Farbe für die Anzeige des Teils* fest.

`part = Color([_colorspec_], [_part_])`

Beachten Sie, dass im Gegensatz zu den anderen Funktionen, die Definition des Teils in diesem Fall an zweiter Stelle steht.

`_colorspec_`

Drei (0-1.0) RGB-Werte und Deckkraft. [R,G,B,A]
Zum Beispiel [1,0,0,0.5] für ein Rot mit 50% Deckkraft.

HALColorFlip

Sets the *display color of the part based on a designated HAL BOOL pin state*.

`part = HALColorFlip([_colorspec_], [_colorspec_], [_part_], hal_comp, hal_pin)`

Beachten Sie, dass im Gegensatz zu den anderen Funktionen, die Definition des Teils in diesem Fall an zweiter Stelle steht.

`_colorspec_`

Drei (0-1.0) RGB-Werte und Deckkraft.
Zum Beispiel [1.0,0,0.5] für ein Rot mit 50% Deckkraft.

`hal_comp`

Die *HAL Komponenten Objekt* oder None.
In QtVCP, wenn Sie *System-Pins* direkt lesen, wird das Komponentenargument auf `None` gesetzt.

`hal_pin`

The *name of the BOOL HAL IN pin* that will change the color.
if hal_comp is *None* then this must be the full name of a system pin other wise this is the pin name excluding the component name.

HALColorRGB

Sets the *display color of the part based on a designated HAL UINT pin value*.
The color is decoded from the UINT value. each color is a 0-255 decimal value (shown here in hex)
red = 0xXXXXXXRR
green = 0xXXXXGGXX
blue = 0xXXBBXXXX
combined as 0xXXBBGGRR

part = HALColorRGB([_part_], hal_comp, hal_pin, alpha=1.0)

hal_comp

Die *HAL Komponenten Objekt* oder None.
In QtVCP, wenn Sie *System-Pins* direkt lesen, wird das Komponentenargument auf `None` gesetzt.

hal_pin

The *name of the UINT HAL IN pin* that will change the color.
if hal_comp is *None* then this must be the full name of a system pin other wise this is the pin name excluding the component name.

alpha=

Setzt die Deckkraft. (0-1.0)

Heads-Up Anzeige (engl. display)

Erzeugt ein *Heads-up-Display* in der Vismach-GUI, um Elemente wie Achsenpositionen, Titel oder Meldungen anzuzeigen.

myhud = Hud()

```
myhud = Hud()
myhud.show("Mill_XYZ")
```

HAL Heads-Up Display

Eine erweiterte Version des Hud, mit der HAL-Pins angezeigt werden können:

myhud = HalHud()

```
myhud = HalHud()
myhud.display_on_right()
myhud.set_background_color(0,.1,.2,0)
myHud.set_text_color(1,1,1)
myhud.show_top("Mill_XYZ")
myhud.show_top("------------")
myhud.add_pin('axis-x: ',"{:10.4f}","axis.x.pos-cmd")
myhud.add_pin('axis-y: ',"{:10.4f}","axis.y.pos-cmd")
myhud.add_pin('axis-z: ',"{:10.4f}","axis.z.pos-cmd")
myhud.show("-------------")
```

Einige der verfügbaren HalHUD Funktionen:

- set_background_color(red, green, blue, alpha)
- add_pin(text, format, pinname)
- set_text_color(red, green, blue)

HideCollection

```
 HideCollection ist ein Container, der einen HAL-Pin verwendet, um die Anzeige der
enthaltenen Teile zu steuern. +
Eine Logik hoch auf dem HAL-Pin verbirgt die enthaltenen Teile.
```

```
comp.newpin("hide-chuck", hal.Type.BOOL, hal.Dir.IN)
# <snip make a machine with an A axis chuck assembly>
chuckassembly = HideCollection([chuckassembly],comp,'hide-chuck')
# also can be used like this
chuckassembly = HideCollection([chuckassembly],None,'myvismach.hide-chuck')
```

Plot-Farbe basierend auf Bewegungstyp (engl. motion type)

Wenn Sie verschiedene Bewegungen (engl. motions) in unterschiedlichen Farben darstellen möchten, müssen Sie zusätzlichen Python-Code hinzufügen.

Fügen Sie dies am Anfang der Datei hinzu:

```
from qtvcp.core import Status
STATUS = Status()
```

und dies zu der Window-Klasse:

```
STATUS.connect('motion-type-changed', lambda w, data: v.choosePlotColor(data))

# auskommentieren, um die feed Farbe zu ändern und alle Farben auf dem Terminal
ausgegeben zu sehen
#v.setColorsAttribute('FEED',(0,1,0))
#print(v.colors)
```

Du kannst Farben zu DEFAULT, FEED, TRAVERSE, ARC, PROBE, ROTARYINDEX, TOOLCHANGE setzen mit setColorsAttribute().

Capture

- `tooltip = Capture()`
  Man kann sich dies als ein unsichtbares Bauteil vorstellen, das am Werkzeugspitzpunkt angebracht werden muss, um Position und Orientierung des Werkzeugkoordinatensystems zu verfolgen. Tatsächlich handelt es sich um eine Transformationsmatrix, die während der Bewegung des Modells fortlaufend aktualisiert wird.

- `work = Capture()`
  Entsprechend wie oben, jedoch am Arbeitstisch angebracht, um das Werkstückkoordinatensystem zu verfolgen.

main

Dies ist der Befehl, der alles in Gang setzt, die Anzeige erstellt usw., wenn er direkt aus Python aufgerufen wird.
Normalerweise wird diese Datei von QtVCP importiert und das `window()` Objekt wird instanziiert und in einen anderen Bildschirm eingebettet.

`main(model, tooltip, work, size=10, hud=myhud, rotation_vectors=None, lat=0, lon=0)`

`_model_`

Sollte eine Sammlung sein, die alle Maschinenteile enthält.

tooltip* und *work

Muss durch `Capture()` erzeugt werden, um den Backplot zu zeichnen, bei dem im Wesentlichen die Werkzeugspitzenposition im Werkstückkoordinatensystem dargestellt wird. Ein Beispiel dafür, wie die Werkzeugspitze mit einem Werkzeug und dieses mit dem Modell verbunden wird, findet sich in `mill_xyz.py`.

`_size_`

Legt die Ausdehnung des in der Ausgangsansicht dargestellten Volumens fest.

`_hud_`

bezieht sich auf eine Head-up-Anzeige.

"*rotation_vectors*" oder "*lat, lon*"

Kann verwendet werden, um den ursprünglichen Blickwinkel festzulegen. Es ist ratsam, zu tun, wie die Standard-Anfangsstandpunkt ist eher wenig hilfreich von unmittelbar über Kopf.

12.10.10. Hinweise

Erstellen Sie für Konstruktionszwecke eine Achsenursprungsmarkierung, um Teile relativ dazu sehen zu können. Sie können diese entfernen, wenn Sie fertig sind.

```
# Erstellen von Achsen-Ursprungs-Markierungen
X = CylinderX(-500,1,500,1)
X = Color([1, 0, 0, 1], [X])
Y = CylinderY(-500,1,500,1)
Y = Color([0, 1, 0, 1], [Y])
Z = CylinderZ(-500,1,500,1)
Z = Color([0, 0, 1, 1], [Z])
origin = Collection([X,Y,Z])
```

Fügen Sie es der Window-Klassensammlung hinzu, damit es nie vom Ursprung verschoben wird.

```
v.model = Collection([origin, model, world])
```

Beginnen Sie an der Schneidspitze (engl. cutting tip) und arbeiten Sie sich zurück. Fügen Sie jede Auflistung dem Modell am Ursprung hinzu, und führen Sie das Skript aus, um den Speicherort zu bestätigen. Drehen / Übersetzen und Ausführen des Skripts zur erneuten Bestätigung.

12.10.11. Grundstruktur eines `QtVismach`-Skripts

```
# imports
import hal
from qtvcp.lib.qt_vismach.qt_vismach import *

# import Status for motion type messages
from qtvcp.core import Status
STATUS = Status()

# create HAL pins here if needed
#c = hal.component("samplegui")
#c.newpin("joint0", hal.Type.REAL, hal.Dir.IN)

# Erstellen des Bodens, des Werkzeugs und des Werkstücks
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()

# Das Modell aufbauen und zusammensetzen
part1 = Collection([Box(-6,-3,94,6,3,100)])
part1 = Color([1,1,1,1],[part1])
part1 = HalRotate([part1],None,"joint.0.pos-fb",360,0,0,1)
part1 = Translate([dogs],-1,49,0)

# ein Top-Level-Modell erstellen
model = Collection([base, saddle, head, carousel])

# wir wollen entweder in QtVCP einbetten oder direkt mit PyQt5 anzeigen
# also ein Fenster bauen, um das Modell anzuzeigen

class Window(QWidget):

    def __init__(self):
        super(Window, self).__init__()
        self.glWidget = GLWidget()
        v = self.glWidget
        v.set_latitudelimits(-180, 180)

        world = Capture()

        # unkommentiert lassen, wenn es ein HUD gibt
        # HUD muss wissen, wo es zeichnen soll
        #v.hud = myhud
        #v.hud.app = v

        # update plot Farbe basierenden auf motion Typ
        STATUS.connect('motion-type-changed', lambda w, data: v.choosePlotColor(data))
```

```
        # auskommentieren, um die Vorschub-Farbe zu ändern
        #v.setColorsAttribute('FEED',(0,1,0))
        # und alle Farben auf dem Terminal auszugeben
        #print(v.colors)

        v.model = Collection([model, world])
        size = 600
        v.distance = size * 3
        v.near = size * 0.01
        v.far = size * 10.0
        v.tool2view = tooltip
        v.world2view = world
        v.work2view = work

        mainLayout = QHBoxLayout()
        mainLayout.addWidget(self.glWidget)
        self.setLayout(mainLayout)

# Wenn Sie diese Datei direkt aus Python3 aufrufen, wird ein PyQt5-Fenster angezeigt.
# das sich gut eignet, um die Teile der Baugruppe zu bestätigen.

if __name__ == '__main__':
    main(model, tooltip, work, size=600, hud=None, lat=-75, lon=215)
```

12.10.12. Integrierte Vismach-Beispielpanels

QtVCP builtin Vismach Panels

12.11. QtVCP: Erstellung benutzerdefinierter Widgets

12.11.1. Übersicht

Die Erstellung von benutzerdefinierten Widgets ermöglicht es, **den Qt Designer-Editor zu verwenden, um ein benutzerdefiniertes Widget zu platzieren**, *anstatt dies manuell in einer Handler-Datei zu tun.*

Ein nützliches benutzerdefiniertes Widget wäre eine großartige Möglichkeit, zu LinuxCNC beizutragen.

Widgets

Widget ist der *allgemeine Name für die UI-Objekte* wie Buttons und Beschriftungen in PyQt.

Es gibt auch **spezielle Widgets für LinuxCNC** zur Erleichterung der Integration.

Alle diese Widgets können *mit Qt Designer Editor* platziert werden - so dass man *das Ergebnis* sehen kann, bevor man das Panel in LinuxCNC lädt.

Qt Designer

Qt Designer ist ein *WYSIWYG (What You See is What You Get) Editor zum Platzieren von PyQt-Widgets.*

Die ursprüngliche Absicht war es, die grafischen Widgets für Programme zu erstellen.
Wir nutzen es, um **Bildschirme und Panels für LinuxCNC** zu bauen.

In Qt Designer, auf der linken Seite des Editors, finden Sie **drei Kategorien von LinuxCNC Widgets**:

- *Nur HAL Widgets*.
- *LinuxCNC-Controller-Widgets*.
- *Dialog-Widgets*.

Damit Qt Designer benutzerdefinierte Widgets zu seinem Editor *hinzufügen* kann, muss ein **Plugin** zum richtigen Ordner hinzugefügt werden.

Initialisierungsprozess

QtVCP macht *extra setup* für **Widgets, die Spezialisierungen** (Unterklassen) **von `_HALWidgetBase`** sind, auch bekannt als "HAL-ified" Widgets.

Dies umfasst:

- Injizieren *wichtiger Variablen,*
- Aufruf einer *extra Setup-Funktion*
- Aufruf einer *schließenden Aufräumfunktion* beim Herunterfahren.

Diese Funktionen werden nicht aufgerufen, wenn der Qt Designer Editor die Widgets anzeigt.

Wenn QtVCP einen Bildschirm aus der *.ui*-Datei erstellt:

1. Es wird nach allen HAL-ifizierten Widgets gesucht.
2. Er findet das Widget `ScreenOptions`, um Informationen zu sammeln, die es in die anderen Widgets einspeisen muss
3. Er instanziiert jedes Widget und ruft, wenn es ein HAL-ifiziertes Widget ist, die Funktion `hal_init()` auf.
   `hal_init()` ist in der Basisklasse definiert und sie:
   a. Fügt Variablen wie die Einstellungsdatei zu jedem HAL-ifizierten Widget hinzu.
   b. Ruft `+_hal_init()+` für das Widget auf.
   `+_hal_init()+` erlaubt es dem Widget-Designer, Einstellungen vorzunehmen, die den Zugriff auf zusätzliche Variablen erfordern.

Hier finden Sie eine Beschreibung der zusätzlichen Variablen, die in "HAL-ifizierte" Widgets eingefügt werden:

`self.HAL_GCOMP`

Die *HAL-Komponenten-Instanz*

`self.HAL_NAME`

Der Name dieses *Widgets* als String

self.QT_OBJECT_

Diese *Objektinstanz des Widgets*

self.QTVCP_INSTANCE_

Die *übergeordnete Ebene* des Bildschirms

self.PATHS_

Die Instanz der *QtVCP-Pfadbibliothek*

self.PREFS_

Die *optionale Instanz der Einstellungsdatei*

self.SETTINGS_

Die `Qsettings` *Objektinstanz*

Aufräum-Prozess

Wenn QtVCP geschlossen wird, ruft es die **`+_hal_cleanup()+`** Funktion *auf allen HAL-ifizierten Widgets* auf.

Die Basisklasse erstellt eine leere Funktion `+_hal_cleanup()+`, die in der Unterklasse des benutzerdefinierten Widgets neu definiert werden kann.

Damit können Sie z. B. Präferenzen aufzeichnen usw.

Diese Funktion wird nicht aufgerufen, wenn der Qt Designer Editor die Widgets anzeigt.

12.11.2. Benutzerdefinierte HAL-Widgets

HAL-Widgets sind die einfachsten, um ein Beispiel zu zeigen.
Die Datei `qtvcp/widgets/simple_widgets.py` enthält viele reine HAL-Widgets.

Sehen wir uns einen Ausschnitt aus `simple_widgets.py` an:

Im Abschnitt "Imports"

Hier importieren wir die Bibliotheken, die unsere Widget-Klasse benötigt.

```
#!/usr/bin/env python3

################################
# Imports
################################
from PyQt5 import QtWidgets                     ①
from qtvcp.widgets.widget_baseclass \
    import _HalWidgetBase, _HalSensitiveBase    ②
import hal                                      ③
```

In diesem Fall benötigen wir Zugang zu:

① Die QtWidgets-Bibliothek von PyQt,

② LinuxCNCs HAL-Bibliothek, und

③ QtVCP's widget `baseclass` 's **`_HalSensitiveBase`** für *automatisches HAL-Pin-Setup* und zum *Deaktivieren/Aktivieren des Widgets* (auch bekannt als Eingangsempfindlichkeit).
Es gibt auch `_HalToggleBase`, und `_HalScaleBase` Funktionen in der Bibliothek.`_HalToggleBase`, und `_HalScaleBase`.

Im Abschnitt WIDGET

Hier ist ein *angepasstes Widget* (engl. custom widget), das auf dem **`QGridLayout`** Widget von PyQt basiert.

`QGridLayout` erlaubt es,:

- *Objekte in einem Raster zu positionieren.*
- *Alle darin enthaltenen Widgets zu aktivieren/deaktivieren*, basierend auf einem **HAL-Pin-Status**.

```
######################
# WIDGET
######################

class Lcnc_GridLayout(QtWidgets.QWidget, _HalSensitiveBase):    ①
    def __init__(self, parent = None):                          ②
        super(GridLayout, self).__init__(parent)                ③
```

Zeile für Zeile:

① This defines the *class name* and the *libraries it inherits from*.
This class, named `Lcnc_GridLayout`, inherits the functions of `QWidget` and `+_HalSensitiveBase+`.
`+_HalSensitiveBase+` is *subclass* of **`+_HalWidgetBase+`**, the *base class of most QtVCP widgets*, meaning it has all the functions of `+_HalWidgetBase+` plus the functions of `+_HalSensitiveBase+`.
It adds the function to make the widget be enabled or disabled based on a HAL input BOOL pin.

② Dies ist die Funktion, die *aufgerufen wird, wenn das Widget zum ersten Mal erstellt wird* (d.h. instanziiert wird) - das ist ziemlich standardmäßig.

③ Diese Funktion initialisiert die **`Super`-Klassen** unseres Widgets.
Super" bedeutet einfach die *vererbten Basisklassen*, d.h. `QWidget` und `_HalSensitiveBase`.
Außer dem Namen des Widgets wird sich nichts ändern.

12.11.3. Benutzerdefinierte Controller-Widgets mit `STATUS`

Widget, die mit LinuxCNC-Controller interagieren sind nur ein wenig komplizierter und sie erfordern einige *extra Bibliotheken*.

In diesem reduzierten Beispiel werden wir Eigenschaften hinzufügen, die im Qt Designer geändert werden können.

Dieses LED-Anzeige-Widget reagiert auf wählbare Zustände der LinuxCNC-Steuerung.

```
#!/usr/bin/env python3

##############################
# Imports
##############################
from PyQt5.QtCore import pyqtProperty
from qtvcp.widgets.led_widget import LED
from qtvcp.core import Status

############################################
# **** instantiate libraries section **** #
############################################
STATUS = Status()

########################################
# custom widget class definition
########################################
class StateLED(LED):
    def __init__(self, parent=None):
        super(StateLED, self).__init__(parent)
        self.has_hal_pins = False
        self.setState(False)
        self.is_estopped = False
        self.is_on = False
        self.invert_state = False

    def _hal_init(self):
        if self.is_estopped:
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
            STATUS.connect('state-estop-reset', lambda w:self._flip_state(False))
        elif self.is_on:
            STATUS.connect('state-on', lambda w:self._flip_state(True))
            STATUS.connect('state-off', lambda w:self._flip_state(False))

    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)

    ######################################################################
    # Qt Designer properties setter/getters/resetters
    ######################################################################

    # invert status
    def set_invert_state(self, data):
        self.invert_state = data
    def get_invert_state(self):
        return self.invert_state
    def reset_invert_state(self):
        self.invert_state = False

    # machine is estopped status
    def set_is_estopped(self, data):
        self.is_estopped = data
    def get_is_estopped(self):
        return self.is_estopped
```

```
    def reset_is_estopped(self):
        self.is_estopped = False

    # machine is on status
    def set_is_on(self, data):
        self.is_on = data
    def get_is_on(self):
        return self.is_on
    def reset_is_on(self):
        self.is_on = False

    ########################################
    # Qt Designer properties
    ########################################
    invert_state_status = pyqtProperty(bool, get_invert_state, set_invert_state,
reset_invert_state)
    is_estopped_status = pyqtProperty(bool, get_is_estopped, set_is_estopped,
reset_is_estopped)
    is_on_status = pyqtProperty(bool, get_is_on, set_is_on, reset_is_on)
```

Im Abschnitt "Imports"

Hier importieren wir die Bibliotheken, die unsere Widget-Klasse benötigt.

```
#!/usr/bin/env python3

################################
# Imports
################################
from PyQt5.QtCore import pyqtProperty       ①
from qtvcp.widgets.led_widget import LED    ②
from qtvcp.core import Status               ③
```

Wir importieren

① `pyqtProperty`, damit wir mit dem Qt Designer-Editor interagieren können,

② `LED`, weil unser benutzerdefiniertes Widget darauf basiert,

③ `Status` weil es uns Statusmeldungen von LinuxCNC gibt.

Im Abschnitt *Bibliotheken instanziieren*

Hier erstellen wir die Bibliotheksinstanz `Status`:

```
############################################
# **** instantiate libraries section **** #
############################################
STATUS = Status()
```

Typischerweise haben wir die Bibliothek *außerhalb der Widget-Klasse* instanziiert, so dass der Verweis auf sie **global** ist - was bedeutet, dass Sie nicht `self.` davor verwenden müssen.

Konventionell verwenden wir *große* Buchstaben im Namen für globale Verweise.

Im Abschnitt "Benutzerdefinierte Widget-Klassendefinition"

Dies ist das Herzstück unseres benutzerdefinierten Widgets.

Klassendefinition und Instanzinitialisierungsfunktion

```
class StateLed(LED):                              ①
    def __init__(self, parent=None):              ②
        super(StateLed, self).__init__(parent)    ③
        self.has_hal_pins = False                 ④
        self.setState(False)                      ⑤
        self.is_estopped = False
        self.is_on = False
        self.invert_state = False
```

① Definiert den **Namen** *unseres benutzerdefinierten Widgets* und von welcher anderen *Klasse es erbt*. In diesem Fall erben wir `LED` - ein QtVCP-Widget, das eine Statusleuchte darstellt.

② Typisch für die meisten Widgets - wird aufgerufen, wenn das Widget zum ersten Mal erstellt wird.

③ Typisch für die meisten Widgets - ruft den Initialisierungscode des übergeordneten (Super-)Widgets auf.

Dann setzen wir einige Attribute:

④ Geerbt von `Lcnc_Led` - wir setzen ihn hier, damit kein HAL-Pin erstellt wird.

⑤ Geerbt von `Lcnc_led` - wir setzen es, um sicherzustellen, dass die LED aus ist.

Die anderen Attribute sind für die auswählbaren Optionen unseres Widgets.

Die HAL-Initialisierungsfunktion des Widgets

```
    def _hal_init(self):
        if self.is_estopped:
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
            STATUS.connect('state-estop-reset', lambda w:self._flip_state(False))
        elif self.is_on:
            STATUS.connect('state-on', lambda w:self._flip_state(True))
            STATUS.connect('state-off', lambda w:self._flip_state(False))
```

Diese Funktion verbindet `STATUS` (LinuxCNC-Statusmeldungsbibliothek) mit unserem Widget, so dass die LED je nach dem gewählten Status des Controllers ein- oder ausgeschaltet wird.

Wir haben zwei Zustände, zwischen denen wir wählen können: `is_estopped` oder `is_on`.
Je nachdem, welcher Zustand aktiv ist, wird unser Widget mit den entsprechenden STATUS-Meldungen verbunden.

`+_hal_init()+` wird _auf jedem Widget aufgerufen, das `+_HalWidgetBase+` erbt, wenn QtVCP zum ersten Mal den Bildschirm aufbaut.
Sie fragen sich vielleicht, warum sie bei diesem Widget aufgerufen wird, da wir `+_HalWidgetBase+` nicht in unserer Klassendefinition haben (`class Lcnc_State_Led(Lcnc_Led):`) - es wird aufgerufen,

weil `Lcnc_Led +_HalWidgetBase+` erbt.

In dieser Funktion haben Sie Zugang zu einigen zusätzlichen Informationen (obwohl wir sie in diesem Beispiel nicht verwenden):

`self.HAL_GCOMP`

Die *HAL-Komponenten-Instanz*

`self.HAL_NAME`

Der Name dieses *Widgets* als String

`self.QT_OBJECT_`

dieses *Widgets PyQt-Objekt als Instanz*

`self.QTVCP_INSTANCE_`

Die *übergeordnete Ebene* des Bildschirms

`self.PATHS_`

Die _Instanz der Pfadbibliothek von QtVCP

`self.PREFS_`

die *Instanz einer optionalen Präferenzdatei*

`self.SETTINGS_`

das `Qsettings` *Objekt*

Wir könnten diese Informationen nutzen, um HAL-Pins zu erstellen oder Bildpfade nachzuschlagen usw.

```
STATUS.connect('state-estop', lambda w:self._flip_state(True))
```

Schauen wir uns diese Zeile genauer an:

- `STATUS` ist ein sehr verbreitetes Thema beim Erstellen von Widgets.
  `STATUS` verwendet das `GObject`-Nachrichtensystem, um Nachrichten an Widgets zu senden, die sich dafür registrieren.
  Diese Zeile ist der Registrierungsprozess.
- `state-estop` ist die Nachricht, auf die wir hören und auf die wir reagieren wollen. Es sind viele Nachrichten verfügbar.
- lambda w:self._flip_state(True)` ist das, was passiert, wenn die Nachricht abgefangen wird.
  Die Lambda-Funktion akzeptiert die Widget-Instanz (`w`), die GObject ihr sendet und ruft dann die Funktion `self._flip_state(True)` auf.
  Lambda wurde verwendet, um das (`w`) Objekt vor dem Aufruf der Funktion `self._flip_state` zu strippen.
  Es erlaubte auch die Verwendung, um `self._flip_state()` den Zustand True zu senden.

```
    def _flip_state(self, data):
            if self.invert_state:
```

```
                data = not data
            self.change_state(data)
```

Dies ist die Funktion, die den Zustand der LED tatsächlich umschaltet.
Sie wird aufgerufen, wenn die entsprechende STATUS-Meldung akzeptiert wird.

```
STATUS.connect('current-feed-rate', self._set_feedrate_text)
```

Die aufgerufene Funktion sieht wie folgt aus:

```
    def _set_feedrate_text(self, widget, data):
```

in dem das Widget und alle Daten von der Funktion akzeptiert werden müssen.

Im Abschnitt *Designer Properties Setter/Getters/Resetters*

```
    ##########################################################################
    # Qt Designer properties setter/getters/resetters
    ##########################################################################

    # invert status
    def set_invert_state(self, data):
        self.invert_state = data
    def get_invert_state(self):
        return self.invert_state
    def reset_invert_state(self):
        self.invert_state = False

    # machine is estopped status
    def set_is_estopped(self, data):
        self.is_estopped = data
    def get_is_estopped(self):
        return self.is_estopped
    def reset_is_estopped(self):
        self.is_estopped = False

    # machine is on status
    def set_is_on(self, data):
        self.is_on = data
    def get_is_on(self):
        return self.is_on
    def reset_is_on(self):
        self.is_on = False
```

Dies ist *die Art und Weise, wie Qt Designer die Attribute des Widgets setzt*.
Dies *kann auch direkt* im Widget aufgerufen werden.

Im Abschnitt "Designereigenschaften"

```
    ######################################
    # Qt Designer properties
```

```
    ######################################
    invert_state_status = pyqtProperty(bool, get_invert_state, set_invert_state,
reset_invert_state)
    is_estopped_status = pyqtProperty(bool, get_is_estopped, set_is_estopped,
reset_is_estopped)
    is_on_status = pyqtProperty(bool, get_is_on, set_is_on, reset_is_on)
```

Dies ist die **Registrierung von Eigenschaften in Qt Designer**.

Der **Eigenschaftsname**:

- ist der *Text, der in Qt Designer* verwendet wird,
- *kann nicht mit den Attributen identisch sein*, die sie darstellen.

Diese Eigenschaften werden im Qt Designer in der Reihenfolge angezeigt, in der sie hier erscheinen.

12.11.4. Benutzerdefinierte Controller-Widgets mit Aktionen

Hier ist ein Beispiel für ein Widget, welches das Benutzerreferenzsystem festlegt.

Es ändert sich:

- den Zustand der Maschinensteuerung mit Hilfe der Bibliothek ACTION,
- ob die Schaltfläche mit Hilfe der STATUS-Bibliothek angeklickt werden kann oder nicht.

```
import os
import hal

from PyQt5.QtWidgets import QWidget, QToolButton, QMenu, QAction
from PyQt5.QtCore import Qt, QEvent, pyqtProperty, QBasicTimer, pyqtSignal
from PyQt5.QtGui import QIcon

from qtvcp.widgets.widget_baseclass import _HalWidgetBase
from qtvcp.widgets.dialog_widget import EntryDialog
from qtvcp.core import Status, Action, Info

# Instanziieren Sie die Bibliotheken mit einer globalen Referenz.
# STATUS gibt uns Statusmeldungen von LinuxCNC
# INFO enthält INI-Details
# ACTION gibt Befehle an LinuxCNC
STATUS = Status()
INFO = Info()
ACTION = Aktion()

class SystemToolButton(QToolButton, _HalWidgetBase):
    def __init__(self, parent=None):
        super(SystemToolButton, self).__init__(parent)
        self._joint = 0
        self._last = 0
        self._block_signal = False
        self._auto_label_flag = True
        SettingMenu = QMenu()
        for system in('G54', 'G55', 'G56', 'G57', 'G58', 'G59', 'G59.1', 'G59.2', 'G59.3
```

```
'):

            Button = QAction(QIcon('exit24.png'), system, self)
            Button.triggered.connect(self[system.replace('.','_')])
            SettingMenu.addAction(Button)

        self.setMenu(SettingMenu)
        self.dialog = EntryDialog()

    def _hal_init(self):
        if not self.text() == '':
            self._auto_label_flag = False
        def homed_on_test():
            return (STATUS.machine_is_on()
                    and (STATUS.is_all_homed() or INFO.NO_HOME_REQUIRED))

        STATUS.connect('state-off', lambda w: self.setEnabled(False))
        STATUS.connect('state-estop', lambda w: self.setEnabled(False))
        STATUS.connect('interp-idle', lambda w: self.setEnabled(homed_on_test()))
        STATUS.connect('interp-run', lambda w: self.setEnabled(False))
        STATUS.connect('all-homed', lambda w: self.setEnabled(True))
        STATUS.connect('not-all-homed', lambda w, data: self.setEnabled(False))
        STATUS.connect('interp-paused', lambda w: self.setEnabled(True))
        STATUS.connect('user-system-changed', self._set_user_system_text)

    def G54(self):
        ACTION.SET_USER_SYSTEM('54')

    def G55(self):
        ACTION.SET_USER_SYSTEM('55')

    def G56(self):
        ACTION.SET_USER_SYSTEM('56')

    def G57(self):
        ACTION.SET_USER_SYSTEM('57')

    def G58(self):
        ACTION.SET_USER_SYSTEM('58')

    def G59(self):
        ACTION.SET_USER_SYSTEM('59')

    def G59_1(self):
        ACTION.SET_USER_SYSTEM('59.1')

    def G59_2(self):
        ACTION.SET_USER_SYSTEM('59.2')

    def G59_3(self):
        ACTION.SET_USER_SYSTEM('59.3')

    def _set_user_system_text(self, w, data):
        convert = { 1:"G54", 2:"G55", 3:"G56", 4:"G57", 5:"G58", 6:"G59", 7:"G59.1", 8:
"G59.2", 9:"G59.3"}
        if self._auto_label_flag:
```

```
            self.setText(convert[int(data)])

    def ChangeState(self, joint):
        if int(joint) != self._joint:
            self._block_signal = True
            self.setChecked(False)
            self._block_signal = False
            self.hal_pin.set(False)

    ##############################
    # required class boiler code #
    ##############################

    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

12.11.5. Stylesheet-Eigenschaftsänderungen auf der Grundlage von Ereignissen

Es ist möglich, **Widgets bei Ereignisänderungen neu zu gestalten**.

Sie müssen das Widget explizit _"polieren", damit PyQt den Stil wiederherstellt.
Dies ist eine relativ teure Funktion und sollte daher sparsam verwendet werden.

Dieses Beispiel setzt eine "isHomed"-Eigenschaft basierend auf dem "homed"-Zustand von LinuxCNC und verwendet diese wiederum, um Stylesheet-Eigenschaften zu ändern:

Dieses Beispiel setzt die Eigenschaft isHomed basierend auf LinuxCNC's homed (engl. für referenziert) Zustandsangabe.

```
class HomeLabel(QLabel, _HalWidgetBase):
    def __init__(self, parent=None):
        super(HomeLabel, self).__init__(parent)
        self.joint_number = 0
        # for stylesheet reading
        self._isHomed = False

    def _hal_init(self):
        super(HomeLabel, self)._hal_init()
        STATUS.connect('homed', lambda w,d: self._home_status_polish(int(d), True))
        STATUS.connect('unhomed', lambda w,d: self._home_status_polish(int(d), False))

    # ishomed-Eigenschaft aktualisieren
    # Widget polieren, damit Stylesheet die Änderung der Eigenschaft sieht
    # einige Stylesheets färben den Text bei home/unhome
    def _home_status_polish(self, d, state):
        if self.joint_number = d:
            self.setProperty('isHomed', state)
            self.style().unpolish(self)
            self.style().polish(self)

    # Qproperty getter and setter
    def getisHomed(self):
```

```
        return self._isHomed
    def setisHomed(self, data):
        self._isHomed = data

    # Qproperty
    isHomed = QtCore.pyqtProperty(bool, getisHomed, setisHomed)
```

Hier ist ein Beispiel-Stylesheet zum Ändern der Textfarbe basierend auf dem Home-Status.

In diesem Fall wird jedes Widget, das auf dem obigen HomeLabel-Widget basiert, die Textfarbe ändern. Normalerweise würden Sie bestimmte Widgets mit `HomeLabel #specific_widget_name[homed=true]` auswählen:

```
HomeLabel[homed=true] {
    color: green;
}
HomeLabel[homed=false] {
    color: red;
}
```

12.11.6. Verwenden von Stylesheets zum Ändern benutzerdefinierter Widget-Eigenschaften

```
class Label(QLabel):
    def __init__(self, parent=None):
        super(Label, self).__init__(parent)
        alternateFont0 = self.font

    # Qproperty getter and setter
    def getFont0(self):
        return self.aleternateFont0
    def setFont0(self, value):
        self.alternateFont0(value)
    # Qproperty
    styleFont0 = pyqtProperty(QFont, getFont0, setFont0)
```

Beispiel-Stylesheet, das eine benutzerdefinierte Widget-Eigenschaft festlegt.

```
Label{
    qproperty-styleFont0: "Times,12,-1,0,90,0,0,0,0,0";
}
```

12.11.7. Widget-Plugins

Wir müssen unsere benutzerdefinierten Widgets *registrieren*, damit Qt Designer sie verwenden kann.

Hier sind ein paar typische Beispiele.
Sie müssten zu `qtvcp/plugins/` hinzugefügt werden.
Dann müsste `qtvcp/plugins/qtvcp_plugin.py` angepasst werden, um sie zu *importieren*.

Beispiel für ein Raster (engl. grid)-Layout

```
#!/usr/bin/env python3

from PyQt5 import QtCore, QtGui
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin
from qtvcp.widgets.simple_widgets import Lcnc_GridLayout
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

####################################
# GridLayout
####################################
class LcncGridLayoutPlugin(QPyDesignerCustomWidgetPlugin):
    def __init__(self, parent = None):
        QPyDesignerCustomWidgetPlugin.__init__(self)
        self.initialized = False
    def initialize(self, formEditor):
        if self.initialized:
            return
        self.initialized = True
    def isInitialized(self):
        return self.initialized
    def createWidget(self, parent):
        return Lcnc_GridLayout(parent)
    def name(self):
        return "Lcnc_GridLayout"
    def group(self):
        return "LinuxCNC - HAL"
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('lcnc_gridlayout')))
    def toolTip(self):
        return "HAL enable/disable GridLayout widget"
    def whatsThis(self):
        return ""
    def isContainer(self):
        return True
    def domXml(self):
        return '<widget class="Lcnc_GridLayout" name="lcnc_gridlayout" />\n'
    def includeFile(self):
        return "qtvcp.widgets.simple_widgets"
```

SystemToolbutton Beispiel

```
#!/usr/bin/env python3

from PyQt5 import QtCore, QtGui
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin
from qtvcp.widgets.system_tool_button import SystemToolButton
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

####################################
# SystemToolButton
####################################
```

```
class SystemToolButtonPlugin(QPyDesignerCustomWidgetPlugin):
    def __init__(self, parent = None):
        super(SystemToolButtonPlugin, self).__init__(parent)
        self.initialized = False
    def initialize(self, formEditor):
        if self.initialized:
            return
        self.initialized = True
    def isInitialized(self):
        return self.initialized
    def createWidget(self, parent):
        return SystemToolButton(parent)
    def name(self):
        return "SystemToolButton"
    def group(self):
        return "LinuxCNC - Controller"
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('systemtoolbutton')))
    def toolTip(self):
        return "Button for selecting a User Coordinate System"
    def whatsThis(self):
        return ""
    def isContainer(self):
        return False
    def domXml(self):
        return '<widget class="SystemToolButton" name="systemtoolbutton" />\n'
    def includeFile(self):
        return "qtvcp.widgets.system_tool_button"
```

Erstellen eines Plugins mit einem Dialogfeld "MenuEntry"

Es ist möglich, einen Eintrag in den Dialog einzufügen, der erscheint, wenn Sie mit der rechten Maustaste auf das Widget im Layout klicken.

So können Sie beispielsweise Optionen auf bequemere Weise auswählen.

Dies ist das Plugin, das für *action buttons* verwendet wird.

```
#!/usr/bin/env python3

import sip
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin, \
                QPyDesignerTaskMenuExtension, QExtensionFactory, \
                QDesignerFormWindowInterface, QPyDesignerMemberSheetExtension
from qtvcp.widgets.action_button import ActionButton
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

Q_TYPEID = {
    'QDesignerContainerExtension':     'org.qt-project.Qt.Designer.Container',
    'QDesignerPropertySheetExtension': 'org.qt-project.Qt.Designer.PropertySheet',
    'QDesignerTaskMenuExtension': 'org.qt-project.Qt.Designer.TaskMenu',
    'QDesignerMemberSheetExtension': 'org.qt-project.Qt.Designer.MemberSheet'
```

```
}

####################################
# ActionBUTTON
####################################
class ActionButtonPlugin(QPyDesignerCustomWidgetPlugin):

    # Die Methode __init__() wird nur verwendet, um das Plugin einzurichten und seine
    # initialisierte Variable zu definieren.
    def __init__(self, parent=None):
        super(ActionButtonPlugin, self).__init__(parent)
        self.initialized = False

    # Die Methoden initialize() und isInitialized() erlauben es dem Plugin, # alle
benötigten Ressourcen einzurichten,
    # wobei sichergestellt wird, dass dies nur einmal für jedes
    # Plugin geschehen kann.
    def initialize(self, formEditor):

        if self.initialized:
            return
        manager = formEditor.extensionManager()
        if manager:
            self.factory = ActionButtonTaskMenuFactory(manager)
            manager.registerExtensions(self.factory, Q_TYPEID['
QDesignerTaskMenuExtension'])
        self.initialized = True

    def isInitialized(self):
        return self.initialized

    # Diese Factory-Methode erstellt neue Instanzen unseres benutzerdefinierten Widgets
    def createWidget(self, parent):
        return ActionButton(parent)

    # Diese Methode gibt den Namen der benutzerdefinierten Widget-Klasse zurück.
    def name(self):
        return "ActionButton"

    # Gibt den Namen der Gruppe in der Widgetbox von Qt Designer zurück
    def group(self):
        return "LinuxCNC - Controller"

    # Gibt das Icon zurück
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('actionbutton')))

    # Gibt eine Kurzbeschreibung des Tooltips zurück
    def toolTip(self):
        return "Aktionsschaltflächen-Widget"

    # Gibt eine kurze Beschreibung des benutzerdefinierten Widgets zur Verwendung in
einer
    # "Was ist das?"-Hilfemitteilung für das Widget.
    def whatsThis(self):
        return ""
```

```
    # Gibt True zurück, wenn das benutzerdefinierte Widget als Container für andere
Widgets fungiert;
    def isContainer(self):
        return False

    # Gibt eine XML-Beschreibung einer benutzerdefinierten Widget-Instanz zurück, welche
die
    # Standardwerte für seine Eigenschaften beschreibt.
    def domXml(self):
        return '<widget class="ActionButton" name="actionbutton" />\n'

    # Gibt das Modul zurück, das die benutzerdefinierte Widget-Klasse enthält. Es kann
    # einen Modulpfad enthalten.
    def includeFile(self):
        return "qtvcp.widgets.action_button"


class ActionButtonDialog(QtWidgets.QDialog):

   def __init__(self, widget, parent = None):

      QtWidgets.QDialog.__init__(self, parent)

      self.widget = widget

      self.previewWidget = ActionButton()

      buttonBox = QtWidgets.QDialogButtonBox()
      okButton = buttonBox.addButton(buttonBox.Ok)
      cancelButton = buttonBox.addButton(buttonBox.Cancel)

      okButton.clicked.connect(self.updateWidget)
      cancelButton.clicked.connect(self.reject)

      layout = QtWidgets.QGridLayout()
      self.c_estop = QtWidgets.QCheckBox("Estop Action")
      self.c_estop.setChecked(widget.estop )
      layout.addWidget(self.c_estop)

      layout.addWidget(buttonBox, 5, 0, 1, 2)
      self.setLayout(layout)

      self.setWindowTitle(self.tr("Set Options"))

   def updateWidget(self):

      formWindow = QDesignerFormWindowInterface.findFormWindow(self.widget)
      if formWindow:
          formWindow.cursor().setProperty("estop_action",
              QtCore.QVariant(self.c_estop.isChecked()))
      self.accept()

class ActionButtonMenuEntry(QPyDesignerTaskMenuExtension):

    def __init__(self, widget, parent):
```

```
        super(QPyDesignerTaskMenuExtension, self).__init__(parent)
        self.widget = widget
        self.editStateAction = QtWidgets.QAction(
          self.tr("Set Options..."), self)
        self.editStateAction.triggered.connect(self.updateOptions)

    def preferredEditAction(self):
        return self.editStateAction

    def taskActions(self):
        return [self.editStateAction]

    def updateOptions(self):
        dialog = ActionButtonDialog(self.widget)
        dialog.exec_()

class ActionButtonTaskMenuFactory(QExtensionFactory):
    def __init__(self, parent = None):
        QExtensionFactory.__init__(self, parent)

    def createExtension(self, obj, iid, parent):

        if not isinstance(obj, ActionButton):
            return None
        if iid == Q_TYPEID['QDesignerTaskMenuExtension']:
            return ActionButtonMenuEntry(obj, parent)
        elif iid == Q_TYPEID['QDesignerMemberSheetExtension']:
            return ActionButtonMemberSheet(obj, parent)
        return None
```

12.12. Code-Schnipsel aus der QtVCP-Handler-Datei

12.12.1. Einstellungsdatei Laden/Speichern

Hier erfahren Sie, wie Sie **die Einstellungen beim Start und beim Beenden laden und speichern**.

Voraussetzungen

- Die Option *Preference file* muss im Widget `ScreenOptions` gesetzt werden.
- Der Pfad der Einstellungsdatei muss in der `INI`-Konfiguration festgelegt werden.

Leseeinstellungen zum Zeitpunkt des Starts

Under the Funktion **`def initialized__(self):`** hinzufügen:

```
if self.w.PREFS_:
    # variable name (entry name, default value, type, section name)
    self.int_value = self.w.PREFS_.getpref('Integer_value', 75, int, '
CUSTOM_FORM_ENTRIES')
    self.string_value = self.w.PREFS_.getpref('String_value', 'on', str,
'CUSTOM_FORM_ENTRIES')
```

Schreibeinstellungen beim Schließen

In der Funktion `closing_cleanup__()` hinzufügen:

```
if self.w.PREFS_:
    # variable name (entry name, variable name, type, section name)
    self.w.PREFS_.putpref('Integer_value', self.integer_value, int, 'CUSTOM_FORM_ENTRIES
')
    self.w.PREFS_.putpref('String_value', self.string_value, str, 'CUSTOM_FORM_ENTRIES')
```

12.12.2. `QSettings` zum Lesen/Speichern von Variablen verwenden

Hier wird beschrieben, wie man **Variablen mit den `QSettings`-Funktionen von PyQt lädt und speichert**:

Gute Praktiken

- Benutzen Sie `Group`, um Namen zu organisieren und eindeutig zu halten.
- Berücksichtigung des Wertes `none` der beim Lesen einer Einstellung zurückgegeben wird, wenn diese keinen Eintrag hat.
- Mit der Syntax `or _<default_value>_` werden die Standardwerte für die erste Ausführung festgelegt.

NOTE	Die Datei wird tatsächlich in `~/.config/QtVcp` gespeichert

Beispiel

In diesem Beispiel:

- Wir fügen `or 20` und `or 2.5` als Standardwerte hinzu.
- Die Namen `MyGroupName`, `int_value`, `float_value`, `myInteger`, und `myFloat` sind benutzerdefiniert.
- Under the Funktion `def initialized__(self):` hinzufügen:

```
# Sortiereinstellungen für aufgezeichnete Spalten festlegen
self.SETTINGS_.beginGroup("MeinGruppenname")
self.int_value = self.SETTINGS_.value('myInteger', Typ = int) or 20
self.float_value = self.SETTINGS_.value('myFloat', type = float) or 2.5
self.SETTINGS_.endGroup()
```

- Unter der Funktion `def closing_cleanup__(self):` hinzufügen:

```
# Werte mit QSettings speichern
self.SETTINGS_.beginGroup("MeinGruppenname")
self.SETTINGS_.setValue('myInteger', self.int_value)
self.SETTINGS_.setValue('myFloat', self.float_value)
self.SETTINGS_.endGroup()
```

12.12.3. Hinzufügen eines grundlegenden Stileditors

Die Möglichkeit, **einen Stil auf einem laufenden Bildschirm zu bearbeiten**, ist sehr praktisch.

Importieren Sie das Modul `StyleSheetEditor` *in der* `IMPORT SECTION`*:*

```
from qtvcp.widgets.stylesheeteditor import StyleSheetEditor as SSE
```

Instanziieren Sie das Modul `StyleSheetEditor` *in der* `INSTANTIATE SECTION`*:*

```
STYLEEDITOR = SSE()
```

Erstellen Sie eine Tastenbelegung (engl. `keybinding`*) in der* `INITIALIZE SECTION`*:*

Unter dem *`init. (self, halcomp, widgets, paths):`* Funktion hinzufügen:

```
KEYBIND.add_call('Key_F12','on_keycall_F12')
```

Erstellen Sie die tastengebundene Funktion im `KEYBINDING SECTION`*:*

```
def on_keycall_F12(self,event,state,shift,cntrl):
    if state:
        STYLEEDITOR.load_dialog()
```

12.12.4. Dialog-Eintrag anfordern

QtVCP verwendet 'STATUS'-Nachrichten, um **Informationen aus Dialogen aufzurufen und zurückzugeben**.

Vorgefertigte Dialoge behalten ihre letzte Position im Auge und enthalten Optionen für Fokusschattierung und Ton.

Um *Informationen aus dem Dialog zurückzubekommen*, muss man eine **`allgemeine`** Nachricht `STATUS` verwenden.

Importieren und Instanziieren des `Status` *Moduls im* `IMPORT ABSCHNITT`

```
from qtvcp.core import Status
STATUS = Status()
```

Dies lädt und initialisiert die `Status`-Bibliothek.

Registrierfunktion für `STATUS general` *Nachrichten im INITIALIZE ABSCHNITT*

Unter der `+__init__. (self, halcomp, widgets, paths)+` Funktion:

```
STATUS.connect('general',self.return_value)
```

Damit wird `STATUS` registriert, um die Funktion `self.return_value` aufzurufen, wenn eine allgemeine Nachricht gesendet wird.

Hinzufügen der Funktion zur Abfrage des Eingabedialogs im Abschnitt `GENERAL FUNCTIONS` *(engl. für allgemeine Funktionen)*

```
def request_number(self):
    mess = {'NAME':'ENTRY','ID':'FORM__NUMBER', 'TITLE':'Set Tool Offset'}
    STATUS.emit('dialog-request', mess)
```

Die Funktion

- erstellt ein Python-`dict` mit:
  - **`NAME`** - muss auf die *Dialog-eindeutigen Start Namen* gesetzt werden.
    `NAME` legt fest, welcher Dialog angefordert werden soll.
    `ENTRY` oder `CALCULATOR` ermöglicht die Eingabe von Zahlen.
  - **`ID`** - muss auf einen *eindeutigen Namen gesetzt werden, den die Funktion liefert*. Die "ID" sollte ein eindeutiger Schlüssel sein.
  - **`TITLE`** setzt den Titel des Dialogs.
  - **Beliebige Daten** können dem *dict* hinzugefügt werden. Das Dialogfeld ignoriert sie, sendet sie aber an den Rückgabecode zurück.
- Sendet das *dict* als ***Dialog-Anfrage*** *STATUS* Nachricht

Hinzufügung der Funktion zur Verarbeitung von Nachrichtendaten im Abschnitt "CALLBACKS FROM STATUS".

```
# Verarbeitung der STATUS-Rückmeldung von set-tool-offset
def return_value(self, w, message):
    num = message.get('RETURN')
    id_code = bool(message.get('ID') == 'FORM__NUMBER')
    name = bool(message.get('NAME') == 'ENTRY')
    if id_code and name and num is not None:
        print('The {} number from {} was: {}'.format(name, id_code, num))
```

Dies fängt alle allgemeinen Nachrichten ab, so dass *der Dialogtyp und der ID-Code* überprüft werden müssen, um zu bestätigen, dass es sich um unseren Dialog handelt. In diesem Fall hatten wir einen `ENTRY`-Dialog angefordert und unsere eindeutige ID war `FORM_NUMBER`, also wissen wir jetzt, dass die Nachricht für uns ist. Die Dialoge `ENTRY` oder `CALCULATOR` geben eine Fließkommazahl zurück.

12.12.5. Sprechen Sie eine Startup-Begrüßung

Dazu muss die Bibliothek `espeak` auf dem System installiert sein.

Importieren und Instanziieren des `Status` *im Abschnitt* `IMPORT`

```
from qtvcp.core import Status
STATUS = Status()
```

Ausgabe gesprochener Nachricht in der INITIALIZE SECTION

Unter der *`init. (self, halcomp, widgets, paths)`* Funktion:

```
STATUS.emit('play-alert','SPEAK Bitte denken Sie daran, die Wege zu ölen.')
```

SPEAK ist ein Schlüsselwort: *Alles, was danach kommt, wird ausgesprochen.*

12.12.6. ToolBar-Funktionen

Symbolleisten-Schaltflächen und Untermenüs werden in Qt Designer hinzugefügt, aber der Code, um sie zu aktivieren, wird in der Handler-Datei hinzugefügt. Um ein Untermenü in Qt Designer **hinzuzufügen**:

- Fügen Sie eine "Qaction" hinzu, indem Sie sie in die Symbolleistenspalte eingeben und dann auf das Symbol "+" auf der rechten Seite klicken.
- Dies wird eine Unterspalte hinzufügen, in die Sie einen Namen eingeben müssen.
- Jetzt wird die ursprüngliche `Qaction` stattdessen ein `Qmenu` sein.
- Löschen Sie nun die `Qaction`, die Sie diesem `Qmenu` hinzugefügt haben, das Menü bleibt als Menü erhalten.

In diesem Beispiel gehen wir davon aus, dass Sie eine Symbolleiste mit einem Untermenü und drei Aktionen hinzugefügt haben. Diese Aktionen werden konfiguriert, um Folgendes zu erstellen:

- ein Menü zur Auswahl zuletzt verwendeter Dateien,
- eine Pop-up-Dialog-Aktion,
- eine Aktion zum Beenden des Programms, und
- eine benutzerdefinierte Funktionsaktion.

Der `objectName` der Symbolleistenschaltfläche wird verwendet, um die Schaltfläche bei der Konfiguration zu identifizieren - *deskriptive Namen helfen.*

Klicken Sie mit der rechten Maustaste auf das Menü des Aktionseditors und wählen Sie Bearbeiten. Editieren Sie den Objektnamen, den Text und den Schaltflächentyp für eine geeignete Aktion.

In diesem Beispiel müssen:

- Der Name des Untermenüs (engl. submenu) `menuRecent` lauten,
- Aktionsnamen `actionAbout`, `actionQuit`, `actionMyFunction` sein

Lädt die Bibliothek `toolbar_actions` *in die* `IMPORT SECTION`

```
from qtvcp.lib.toolbar_actions import ToolBarActions
```

Instanziieren Sie das Modul `ToolBarActions` *im Abschnitt* `INSTANTIATE LIBRARY SECTION`

```
TOOLBAR = ToolBarActions()
```

Konfigurieren Sie Untermenüs und Aktionen im Abschnitt "BESONDERE FUNKTIONEN"

Unter der Funktion `def initialized__(self)` hinzufügen:

```
TOOLBAR.configure_submenu(self.w.menuRecent, 'recent_submenu')
TOOLBAR.configure_action(self.w.actionAbout, 'about')
TOOLBAR.configure_action(self.w.actionQuit, 'Quit', lambda d:self.w.close())
TOOLBAR.configure_action(self.w.actionMyFunction, 'My Function', self.my_function)
```

Definieren Sie die Benutzerfunktion in der `GENERAL FUNCTIONS SECTION`

```
def my_function(self, widget, state):
    print('My function State = ()'.format(state))
```

Die Funktion, die aufgerufen werden soll, wenn die Aktionsschaltfläche "Meine Funktion" gedrückt wird.

12.12.7. HAL Pins hinzufügen, die Funktionen aufrufen

Auf diese Weise müssen Sie den *Zustand der Eingangsstifte nicht abfragen*.

Lädt die Bibliothek `Qhal` *in die* `IMPORT SECTION`

```
from qtvcp.core import Qhal
```

Dies dient dem Zugriff auf **QtVCPs HAL-Komponente**. Weitere Informationen: Qhal
Die newPin-Funktion von Qhal gibt ein QPin-Objekt zurück. Weitere Informationen: QPin

Instanziieren von `Qhal` *in der* `INSTANTIATE LIBRARY SECTION`

```
QHAL = Qhal()
```

Hinzufügen einer Funktion, die aufgerufen wird, wenn sich der Zustand des Pins ändert

Vergewissern Sie sich, dass unter der Funktion `initialized__` ein Eintrag ähnlich dem folgenden vorhanden ist:

```
########################################
# Spezielle Funktionen, die von QtVCP aufgerufen werden
########################################

# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    self.pin_cycle_start_in = QHAL.newPin('cycle-start-in',QHAL.HAL_BOOL, QHAL.HAL_IN)
    self.pin_cycle_start_in.pinValueChanged.connect(lambda o,s: self.cycleStart(s))
```

Definieren der Funktion, die durch Pin-Statusänderung im GENERAL FUNCTIONS SECTION aufgerufen wird

```
####################################
# allgemeine (engl. general) functions #
####################################

def cycleStart(self, state):
```

```
    if state:
        tab = self.w.mainTab.currentWidget()
        if  tab in( self.w.tab_auto,  self.w.tab_graphics):
            ACTION.RUN(line=0)
        elif tab == self.w.tab_files:
                self.w.filemanager.load()
        elif tab == self.w.tab_mdi:
            self.w.mditouchy.run_command()
```

Diese Funktion geht davon aus, dass es ein Tab-Widget mit dem Namen `mainTab` gibt, das Tabs mit den Namen `tab_auto`, `tab_graphics`, `tab_filemanager` und `tab_mdi` hat.

Auf diese Weise funktioniert der Zyklusstart-Button je nach angezeigter Registerkarte unterschiedlich.

Dies ist vereinfacht - *Zustandskontrolle und Fehlerverfolgung könnten hilfreich sein.*

12.12.8. Direktes Lesen/Schreiben von System HAL Pins

Manchmal muss ein System-Pin ausgelesen werden, und dafür extra einen HAL-Pin zu erstellen und zu verbinden, ist mehr Aufwand als nötig. Sie können ihn direkt auslesen, ohne eine Verbindung herzustellen.

Hier wird beschrieben, wie Sie einen Pin, einen Parameter oder ein Signal lesen:

```
self.h.hal.get_value('spindle.0.at-speed')

# oder Sie nutzen die Qhal Bibliothek wie folgt
QHAL.getValue('spindle.0.at-speed')
```

So schreiben Sie in einen nicht verbundenen Pin oder ein nicht gesteuertes Signal:

```
self.h.hal.set_p('componentName.pinName','10')
self.h.hal.set_s('componentName.signalName','10')

# oder Sie nutzen die Qhal Biblithek wie folgt:
QHAL.setPin('componentName.pinName', '10')
QHAL.setSignal('componentName.signalName', '10')
```

Die Verwendung von *self.h.hal* oder *QHAL.hal* ermöglicht den Zugriff auf die Funktionen des HAL-Python-Moduls: Python HAL Interface

12.12.9. Hinzufügen eines speziellen prozentualen Max Velocity Sliders

Manchmal möchten Sie **ein Widget erstellen, um etwas zu tun, das nicht eingebaut ist**. Der eingebaute Schieberegler für die maximale Geschwindigkeit wirkt auf Einheiten pro Minute, hier zeigen wir, wie man mit Prozent arbeitet.

Der **`STATUS`** Befehl stellt sicher, dass der Schieberegler sich anpasst, wenn LinuxCNC die aktuelle maximale Geschwindigkeit ändert.

`valueChanged.connect()` *ruft eine Funktion auf, wenn der Schieberegler bewegt wird.*

Fügen Sie im Qt Designer ein `QSlider`-Widget mit dem Namen `mvPercent` hinzu und fügen Sie dann den folgenden Code in die Handler-Datei ein:

```
############################
# SPECIAL FUNCTIONS SECTION #
############################

def initialized__(self):
    self.w.mvPercent.setMaximum(100)
    STATUS.connect('max-velocity-override-changed', \
        lambda w, data: self.w.mvPercent.setValue( \
            (data / INFO.MAX_TRAJ_VELOCITY)*100 \
            )
        )
    self.w.mvPercent.valueChanged.connect(self.setMVPercentValue)

####################
# GENERAL FUNCTIONS #
####################

def setMVPercentValue(self, value):
    ACTION.SET_MAX_VELOCITY_RATE(INFO.MAX_TRAJ_VELOCITY * (value/100.0))
```

12.12.10. Kontinuierlichen Jog ein- und ausschalten

Im Allgemeinen ist die Auswahl des kontinuierlichen Joggens eine Momenttaste, nach der Sie die vorherige Jogging-Stufe auswählen müssen.

Wir werden eine Schaltfläche erstellen, die zwischen kontinuierlichem Joggen und der bereits ausgewählten Schrittweite umschaltet.

Im Qt-Designer:

- Hinzufügen eines `ActionButton` ohne Aktion
- Nennen Sie ihn "btn_toggle_continuous".
- Setzen Sie die Eigenschaft `AbstractButton checkable` auf `True`.
- Setzen Sie die Eigenschaften `ActionButton incr_imperial_number` und `incr_mm_number` auf `0`.
- Verwenden Sie den Slot-Editor von Qt Designer, um das Button-Signal `clicked(bool)` zu verwenden, um die Formular-Handler-Funktion `toggle_continuous_clicked()` aufzurufen. Siehe Using Qt Designer To Add Slots Abschnitt für weitere Informationen.

Dann fügen Sie diesen Code-Schnipsel in die Handler-Datei unter der Funktion `initialized__` ein:

```
# zu diesem Zeitpunkt:
# sind die Widgets instanziiert.
# die HAL-Pins sind gebaut, aber HAL ist nicht bereit
def initialized__(self):
    STATUS.connect('jogincrement-changed', \
```

```
        lambda w, d, t: self.record_jog_incr(d,t) \
        )
    # ein Standardinkrement festlegen, zu dem zurückgeschaltet werden soll
    self.L_incr = 0,01
    self.L_text = "0.01in"
```

Im Abschnitt `ALLGEMEINE FUNKTIONEN` (engl. general functions) hinzufügen:

```
####################
# GENERAL FUNCTIONS #
####################

# Wenn es nicht kontinuierlich ist, zeichnen Sie das letzte Jog-Inkrement auf.
# und deaktiviere die Schaltfläche "kontinuierlich".
def record_jog_incr(self,d, t):
    if d != 0:
        self.L_incr = d
        self.L_text = t
        self.w.btn_toggle_continuous.safecheck(False)
```

Im Abschnitt `CALLBACKS FROM STATUS SECTION` hinzufügen:

```
############################
# CALLBACKS VOM FORMULAR #
############################

def toggle_continuous_clicked(self, state):
    if state:
        # set continuous (call the actionbutton's function)
        self.w.btn_toggle_continuous.incr_action()
    else:
        # reset previously recorded increment
        ACTION.SET_JOG_INCR(self.L_incr, self.L_text)
```

12.12.11. Klassen-Patch Dateimanager-Widget

NOTE

Class Patching (Monkey Patching) ist ein bisschen wie *schwarze Magie* – verwenden Sie es also *nur bei Bedarf*. Das Hauptproblem besteht darin, dass die Funktionen der Widget-Bibliothek während der Entwicklung geändert werden können und die Funktionen dadurch fehlschlagen können.
Das File-Manager-Widget ist dafür vorgesehen, ein ausgewähltes Programm in LinuxCNC zu laden.
Vielleicht möchten Sie jedoch zuerst den Dateinamen ausgeben.

Wir können die Bibliothek "class patchen", um den *Funktionsaufruf umzuleiten*.
Sie können dieses Class Patching innerhalb oder außerhalb der HandlerClass-Instanz durchführen.
Dies ändert, was *self* in der Funktion repräsentiert.
Außerhalb der HandlerClass ist *self* die Instanz der gepatchten Klasse.
Innerhalb der HandlerClass ist *self* die HandlerClass-Instanz.
Dies wirkt sich darauf aus, welche Funktionen/Variablen Sie in der Funktion aufrufen können.

Hier zeigen wir ein Beispiel innerhalb der HandlerClass:
Fügen Sie im `IMPORT SECTION` Folgendes hinzu:

```
from qtvcp.widgets.file_manager import FileManager as FM
```

Hier werden wir:

1. *Behalten Sie einen Verweis auf die ursprüngliche Funktion* (1), damit wir sie weiterhin aufrufen können
2. Leiten Sie die Klasse um, damit sie stattdessen unsere benutzerdefinierte Funktion (2) in der Handler-Datei aufruft.

```
########################################
# Spezielle Funktionen, die von QtVCP aufgerufen werden
########################################

# Um Funktionen in Widgets zu ändern, können wir 'class patch' verwenden.
# Klassenpatching muss vor der Instanziierung der Klasse erfolgen.
def class_patch__(self):
    self.old_load = FM.load # behalte eine Referenz auf die alte Funktion ①
    FM.load = self.our_load # Funktion auf unsere Handle-Datei-Funktion umleiten ②
```

3. Schreiben Sie eine benutzerdefinierte Funktion, um das Original zu ersetzen:
   Diese Funktion muss die **gleiche Signatur wie die Originalfunktion** haben.
   self ist die HandlerClass-Instanz *not* the gepatchte class Instanz.
   In diesem Beispiel werden wir immer noch die ursprüngliche Funktion aufrufen, indem wir den Verweis auf sie verwenden, den wir zuvor aufgezeichnet haben.
   Sie *erfordert, dass das erste Argument die Widget-Instanz ist*, was in diesem Fall `self.w.filemanager` ist (der Name, der im Qt Designer-Editor angegeben wurde).

```
####################
# GENERAL FUNCTIONS #
####################

def our_load(self,fname):
    print(fname)
    self.old_load(self.w.filemanager,fname)
```

Jetzt wird unsere benutzerdefinierte Funktion den Dateipfad in das Terminal ausgeben, bevor die Datei geladen wird. Offensichtlich langweilig, aber es zeigt das Prinzip.

NOTE

Es gibt noch eine andere, etwas andere Methode, die Vorteile haben kann: Sie können *den Verweis auf die ursprüngliche Funktion in der ursprünglichen Klasse speichern.*
Der Trick dabei ist, sicherzustellen, dass der Funktionsname, den Sie zum Speichern verwenden, nicht bereits in der Klasse verwendet wird.
`super__` als Zusatz zum Funktionsnamen wäre eine gute Wahl.
Wir werden das in den eingebauten QtVCP-Widgets nicht verwenden.

```
########################################
```

```
# Spezielle Funktionen, die von QtVCP aufgerufen werden
########################################

# Um Funktionen in Widgets zu ändern, können wir 'class patch' verwenden.
# Klassenpatching muss vor der Instanziierung der Klasse erfolgen.
def class_patch__(self):
    FM.super__load = FM.load # eine Referenz auf die alte Funktion in der
ursprünglichen Klasse behalten
    FM.load = self.our_load # Funktion auf unsere Handle-File-Funktion
umlenken

######################
# GENERAL FUNCTIONS #
######################

def our_load(self,fname):
    print(fname)
    self.w.filemanager.super__load(fname)
```

12.12.12. Widgets programmatisch hinzufügen

In manchen Situationen ist es nur möglich, Widgets mit Python-Code hinzuzufügen, anstatt den Qt Designer-Editor zu verwenden.

Wenn QtVCP-Widgets programmatisch hinzugefügt werden, müssen manchmal *zusätzliche Schritte* unternommen werden.

Hier werden wir eine Spindeldrehzahl-Anzeigeleiste und eine LED für die aktuelle Geschwindigkeit in die Ecke eines Tab-Widgets einfügen. Qt Designer unterstützt das Hinzufügen von Eck-Widgets zu Tabs nicht, PyQt hingegen schon.

Dies ist ein gekürztes Beispiel aus der Handler-Datei von QtAxis screen.

Importieren erforderlicher Bibliotheken

Zunächst müssen wir die benötigten Bibliotheken importieren, sofern sie nicht bereits in der Handler-Datei enthalten sind:

- `QtWidgets` gibt uns Zugriff auf die `QProgressBar`,
- `QColor` ist für die *LED-Farbe*,
- `StateLED` ist die QtVCP-Bibliothek, die zum *Erstellen der Spindel-bei-Geschwindigkeit-LED* verwendet wird,
- `Status` wird verwendet, um *LinuxCNC-Statusinformationen abzufangen*,
- `Info` gibt uns *Informationen über die Maschinenkonfiguration*.

```
############################
# **** IMPORT SECTION **** #
############################

from PyQt5 import QtWidgets
```

```
from PyQt5.QtGui import QColor
from qtvcp.widgets.state_led import StateLED as LED
from qtvcp.core import Status, Info
```

Instanziierung von `Status`*- und* `Info`*-Kanälen*

`STATUS` und `INFO` werden außerhalb der Handler-Klasse initialisiert, so dass es sich um *globale Referenzen* handelt (kein self. vorangestellt):

```
############################################
**** Bibliotheken instanziieren Abschnitt **** #
############################################

STATUS = Status()
INFO = Info()
```

Register `STATUS` *Überwachungsfunktion*

Für die Spindeldrehzahlanzeige müssen wir die aktuelle Spindeldrehzahl kennen. Dazu *registrieren* wir uns mit `STATUS`, um:

- *Erfassen* des `actual-spindle-speed-changed` *Signals*
- *Aufruf* der _Funktion `self.update_spindle()`

```
########################
# **** INITIALIZE **** #
########################
# Ermöglicht den Zugriff auf Widgets aus den QtVCP-Dateien.
# Zu diesem Zeitpunkt sind die Widgets und Hal-Pins noch nicht instanziiert
def __init__(self,halcomp,widgets,paths):
    self.hal = halcomp
    self.w = widgets
    self.PATHS = paths

    STATUS.connect('actual-spindle-speed-changed', \
        lambda w,speed: self.update_spindle(speed))
```

Hinzufügen der Widgets zur Registerkarte

Wir müssen *sicherstellen, dass die Qt Designer Widgets bereits gebaut sind*, bevor wir versuchen, sie zu ergänzen. Zu diesem Zweck fügen wir einen Aufruf der Funktion `self.make_corner_widgets()` hinzu, um unsere zusätzlichen Widgets zum richtigen Zeitpunkt zu erstellen, d.h. unter der Funktion `initialized__()`:

```
########################################
# Spezielle Funktionen, die von Qt Screen aufgerufen werden
########################################

# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    self.make_corner_widgets()
```

Erstellen der Funktionen zum Erstellen von Widgets

Ok, lassen Sie uns die Funktion zum Erstellen der Widgets codieren und sie im Tab-Widget hinzufügen. Wir gehen davon aus, dass es ein mit Designer erstelltes Tab-Widget namens *rightTab* gibt.

Wir gehen davon aus, dass es ein Tab-Widget gibt, das mit Qt Designer gebaut wurde und `rightTab` heißt.

```
########################################
# allgemeine (engl. general) functions #
########################################

def make_corner_widgets(self):
    # make a spindle-at-speed green LED
    self.w.led = LED()                                          ①
    self.w.led.setProperty('is_spindle_at_speed_status',True) ②
    self.w.led.setProperty('color',QColor(0,255,0,255))        ③
    self.w.led.hal_init(HAL_NAME = 'spindle_is_at_speed')      ④

    # make a spindle speed bar
    self.w.rpm_bar = QtWidgets.QProgressBar()                   ⑤
    self.w.rpm_bar.setRange(0, INFO.MAX_SPINDLE_SPEED)          ⑥

    # container
    w = QtWidgets.QWidget()                                     ⑦
    w.setContentsMargins(0,0,0,6)
    w.setMinimumHeight(40)

    # layout
    hbox = QtWidgets.QHBoxLayout()                              ⑧
    hbox.addWidget(self.w.rpm_bar)                              ⑨
    hbox.addWidget(self.w.led)                                  ⑨
    w.setLayout(hbox)

    # den Container zur Ecke des rechten Tab-Widgets hinzufügen
    self.w.rightTab.setCornerWidget(w) ⑩
```

① Dies initialisiert das grundlegende StateLed-Widget und verwendet von da an `self.w.led` als Referenz.

② Da die Zustands-LED für viele Anzeigen verwendet werden kann, müssen wir die Eigenschaft einstellen, die sie als LED für die Spindeldrehzahl kennzeichnet.

③ Dadurch wird sie im eingeschalteten Zustand als grün angezeigt.

④ Dies ist der zusätzliche Funktionsaufruf, der bei einigen QtVCP-Widgets erforderlich ist.
Wenn `HAL_NAME` weggelassen wird, so wird der `Objektname` des Widgets verwendet, sofern es einen gibt.
Es gibt den speziellen Widgets eine Referenz zu:

`self.HAL_GCOMP`

Die *HAL-Komponenten-Instanz*

`self.HAL_NAME`

Der Name dieses *Widgets* als String

self.QT_OBJECT_

dieses *Widgets PyQt-Objekt als Instanz*

self.QTVCP_INSTANCE_

Die *übergeordnete Ebene* des Bildschirms

self.PATHS_

Die _Instanz der Pfadbibliothek von QtVCP

self.PREFS_

die *Instanz einer optionalen Präferenzdatei*

self.SETTINGS_

das `Qsettings` *Objekt*

⑤ Initialisiert einen PyQt5 `QProgressBar`.

⑥ Setzt den maximalen Bereich des Fortschrittsbalkens auf den in der `INI` angegebenen Maximalwert.

⑦ Wir erstellen ein QWidget
Da man nur ein Widget in die Tab-Ecke einfügen kann und wir dort zwei haben wollen, müssen wir beide in einen **Container** einfügen.

⑧ ein QHBoxLayout zum QWidget hinzufügen.

⑨ Dann fügen wir unseren QProgress-Balken und die LED in das Layout ein.

⑩ Schließlich fügen wir das QWidget (mit unserem QProgress-Balken und der LED darin) in die Ecke des Tab-Widgets ein.

Erstellen Sie die Überwachungsfunktion `STATUS`

Jetzt erstellen wir die Funktion, die den `QProgressBar` aktualisiert, wenn `STATUS` die Spindeldrehzahl aktualisiert:

```
####################
# callbacks von STATUS #
####################
def update_spindle(self, data):
    self.w.rpm_bar.setInvertedAppearance(bool(data<0))       ①
    self.w.rpm_bar.setFormat('{0:d} RPM'.format(int(data)))  ②
    self.w.rpm_bar.setValue(abs(data))                       ③
```

① In diesem Fall haben wir uns für die Darstellung von links nach rechts oder von rechts nach links entschieden, je nachdem, ob wir uns im oder gegen den Uhrzeigersinn drehen.

② Dadurch wird die Schrift in der Leiste formatiert.

③ Damit legen Sie die Länge des farbigen Balkens fest.

12.12.13. Objekte periodisch aktualisieren/auslesen

Manchmal muss man **ein Widget aktualisieren oder regelmäßig einen Wert auslesen**, der von den normalen Bibliotheken nicht abgedeckt wird.

Hier aktualisieren wir eine LED auf der Grundlage eines überwachten HAL-Pins alle 100 ms.

Wir nehmen an, dass in der Qt Designer UI Datei eine LED mit dem Namen `led` vorhanden ist.

Laden Sie die `Qhal`-Bibliothek für den Zugriff auf die HAL-Komponente von QtVCP

In der `IMPORT SECTION` hinzufügen:

```
from qtvcp.core import Qhal
```

Instanziiere `Qhal`

Im Abschnitt `INSTANTIATE LIBRARY` hinzufügen:

```
QHAL = Qhal()
```

Fügen Sie nun diese Abschnitte hinzu bzw. ändern Sie sie so, dass sie einen ähnlichen Code enthalten wie dieser:

Registrierung einer Funktion, die im Zeitraum `CYCLE_TIME` aufgerufen wird

Dies geschieht normalerweise alle 100 ms.

```
########################
# **** INITIALIZE **** #
########################
# Ermöglicht den Zugriff auf Widgets aus den QtVCP-Dateien.
# Zu diesem Zeitpunkt sind die Widgets und Hal-Pins noch nicht instanziiert
def __init__(self,halcomp,widgets,paths):
    self.hal = halcomp
    self.w = widgets
    self.PATHS = paths

    # eine Funktion registrieren, die in einer CYCLE_TIME-Periode aufgerufen wird
(normalerweise alle 100 ms)
    STATUS.connect('periodic', lambda w: self.update_periodic())
```

Erstellen Sie die benutzerdefinierte Funktion, die periodisch aufgerufen werden soll

```
####################
# general functions #
####################
def update_periodic(self):
    data = QHAL.getvalue('spindle.0.is-oriented')
    self.w.led.setState(data)
```

12.12.14. Externe Steuerung mit ZMQ

QtVCP kann automatisch ein **ZMQ-Messaging** *einrichten, um Remote-Nachrichten von externen Programmen zu senden und/oder zu empfangen.*

Es verwendet das ZMQ-Meldungsmuster **Veröffentlichen/Abonnieren**.

Wie immer sollte man die **Sicherheit** im Auge behalten, bevor man Programmen eine Schnittstelle für Nachrichtenübermittlung einräumt.

Lesen von ZMQ-Nachrichten

Manchmal möchte man den **Bildschirm mit einem separaten Programm steuern**.

Aktivieren des Empfangs von ZMQ-Nachrichten

Im Widget `ScreenOptions` können Sie die Eigenschaft **`use_receive_zmq_option`** auswählen.
Sie können diese Eigenschaft auch direkt *in der Handler-Datei* einstellen, wie in diesem Beispiel.

Wir nehmen an, dass das `ScreenOptions`-Widget in Qt Designer `screen_options` genannt wird:

```
########################
# **** INITIALIZE **** #
########################
# widgets erlaubt den Zugriff auf Widgets aus den QtVCP-Dateien.
# zu diesem Zeitpunkt sind die Widgets und Hal-Pins noch nicht instanziiert
def __init__(self, halcomp,widgets,paths):
    # directly select ZMQ message receiving
    self.w.screen_options.setProperty('use_receive_zmq_option',True)
```

Dies **erlaubt einem externen Programm, Funktionen in der Handler-Datei aufzurufen**.

Hinzufügen einer Funktion, die beim Empfang einer ZMQ-Nachricht aufgerufen wird

Fügen wir eine spezielle Funktion zum Testen hinzu. Sie müssen LinuxCNC von einem Terminal aus starten, um den gedruckten Text zu sehen.

```
####################
# general functions #
####################
def test_zmq_function(self, arg1, arg2):
    print('zmq_test_function called: ', arg1, arg2)
```

Erstellen eines externen Programms, das ZMQ-Nachrichten sendet, die einen Funktionsaufruf auslösen

Hier ist ein Beispiel für ein externes Programm zum Aufruf einer Funktion. Es wechselt jede Sekunde zwischen zwei Datensätzen. Führen Sie dies in einem separaten Terminal von LinuxCNC aus, um die gesendeten Nachrichten zu sehen.

```
#!/usr/bin/env python3
from time import sleep

import zmq
import json

context = zmq.Context()
socket = context.socket(zmq.PUB)
socket.bind("tcp://127.0.0.1:5690")
topic = b'QtVCP'
```

```
# prebuilt message 1
# Erstellt ein dict  der aufzurufenden Funktion plus Argumenten
x = {                                ①
  "FUNCTION": "test_zmq_function",
  "ARGS": [True,200]
}
# Konvertiert zu JSON Objekt
m1 = json.dumps(x)

# prebuild message 2
x = {                                ①
  "FUNCTION": "test_zmq_function",
  "ARGS": [False,0],
}
# Konvertiert zu JSON Objekt
m2 = json.dumps(x)

if __name__ == '__main__':
    while True:
        print('send message 1')
        socket.send_multipart([topic, bytes((m1).encode('utf-8'))])
        sleep(ms(1000))

        print('send message 2')
        socket.send_multipart([topic, bytes((m2).encode('utf-8'))])
        sleep(ms(1000))
```

① Legen Sie die **aufzurufende Funktion** und die **zu sendenden Argumente** auf diese Funktion fest.

Sie müssen die *Signatur* der Funktion kennen, die Sie aufrufen möchten. Beachten Sie auch, dass die *Nachricht in ein JSON-Objekt* umgewandelt wird. Das liegt daran, dass ZMQ Byte-Nachrichten und keine Python-Objekte sendet. `json` konvertiert Python-Objekte in Bytes und wird beim Empfang wieder zurück konvertiert.

Schreiben von ZMQ-Nachrichten

Sie können auch **mit einem externen Programm vom Bildschirm aus kommunizieren**.

Im Widget `ScreenOptions` können Sie die Eigenschaft **`use_send_zmq_message`** auswählen. Sie können diese Eigenschaft auch direkt *in der Handler-Datei* einstellen, wie in diesem Beispiel.

Wir nehmen an, dass das `ScreenOptions`-Widget in Qt Designer `screen_options` genannt wird:

Senden von ZMQ-Nachrichten aktivieren

```
########################
# **** INITIALIZE **** #
########################
# 'widgets' erlaubt den Zugriff auf Widgets aus den QtVCP-Dateien.
# zu diesem Zeitpunkt sind die Widgets und Hal-Pins noch nicht instanziiert
def __init__(self, halcomp,widgets,paths):
    # directly select ZMQ message sending
    self.w.screen_options.setProperty('use_send_zmq_option',True)
```

Dies ermöglicht das Senden von Nachrichten an ein separates Programm.
Welche Nachricht gesendet wird, hängt davon ab, was das externe Programm erwartet.

Erstellen einer Funktion zum Senden von ZMQ-Nachrichten

Lassen Sie uns eine spezielle Funktion zum Testen hinzufügen.
Sie müssen LinuxCNC von einem Terminal aus starten, um den gedruckten Text zu sehen.
Außerdem muss etwas hinzugefügt werden, um diese Funktion aufzurufen, wie z.B. ein Tastenklick.

```
######################
# allgemeine Funktionen #
######################
def send_zmq_message(self):
    # This could be any Python object JSON can convert
    message = {"name": "John", "age": 30}
    self.w.screen_options.send_zmq_message(message)
```

Verwenden oder erstellen Sie ein Programm, das ZMQ-Nachrichten empfangen kann

Hier ist ein Beispielprogramm, das die Nachricht empfängt und auf dem Terminal ausgibt:

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# Definition des Sockets mit Hilfe des "Context".
sock = context.socket(zmq.SUB)

# Definieren des Abonnements und der zu akzeptierenden Nachrichten ohne Einschränkung der
Themen.
topic = "" # alle Themen
sock.setsockopt_string(zmq.SUBSCRIBE, topic)
sock.connect("tcp://127.0.0.1:5690")

while True:
    topic, message = sock.recv_multipart()
    print('{} sent message:{}'.format(topic,json.loads(message)))
```

12.12.15. Senden von Nachrichten an die Statusleiste oder Desktop-Benachrichtigungsdialoge

Es gibt mehrere Möglichkeiten, dem Benutzer **Informationen zu übermitteln**.

Eine **Statusleiste** wird für *short information* verwendet, um den Benutzer anzuzeigen.

NOTE	Nicht alle Bildschirme haben eine Statusleiste.

Beispiel für die Nutzung der Statusleiste

```
self.w.statusbar.showMessage(message, timeout * 1000)
```

`timeout` ist in Sekunden und wir nehmen an, dass "statusbar" der Name des Qt Designer Widgets ist.

Sie können auch die 'Status'-Bibliothek verwenden, um eine Nachricht an die 'notify'-Bibliothek zu senden, wenn diese aktiviert ist (normalerweise im 'ScreenOptions'-Widget eingestellt): Dadurch wird die Nachricht an die Statusleiste und den **Desktop-Benachrichtigungsdialog** (engl. notify dialog) gesendet.

Die Nachrichten werden auch aufgezeichnet, bis der Benutzer sie mit Hilfe der Bedienelemente löscht. Die Benutzer können alle aufgezeichneten Nachrichten abrufen.

Es gibt mehrere Optionen:

`STATUS.TEMPORARY_MESSAGE`

Anzeigen der Nachricht nur für eine kurze Zeit.

`STATUS.OPERATOR_ERROR`

`STATUS.OPERATOR_TEXT`

`STATUS.NML_ERROR`

`STATUS.NML_TEXT`

Beispiel für das Versenden einer Bedienermeldung:

```
STATUS.emit('error', STATUS.OPERATOR_ERROR, 'message')
```

Sie können Nachrichten über die Bedienernachrichtenfunktionen (engl. operator message functions) von LinuxCNC senden. Diese werden normalerweise vom Benachrichtigungssystem abgefangen, sind also gleich oben. Sie würden auch auf das Terminal gedruckt werden.

```
ACTION.SET_DISPLAY_MESSAGE('MESSAGE')
ACTION.SET_ERROR_MESSAGE('MESSAGE')
```

12.12.16. Fokusänderungen abfangen

Der Fokus wird verwendet, um Benutzeraktionen wie Tastatureingaben auf das richtige Widget zu lenken.

Aktuell fokussiertes Widget abrufen

```
fwidget = QtWidgets.QApplication.focusWidget()
if fwidget is not None:
    print("focus widget class: {} name: {} ".format(fwidget, fwidget.objectName()))
```

Fokussiertes Widget erhalten, wenn sich der Fokus ändert

```
# zu diesem Zeitpunkt:
# sind die Widgets instanziiert.
# die HAL-Pins sind gebaut, aber HAL ist nicht bereit
def initialized__(self):
    QtWidgets.QApplication.instance().event_filter.focusIn.connect(self.focusInChanged)
```

```
#####################################
# allgemeine (engl. general) functions #
#####################################

def focusInChanged(self, widget):
    if isinstance(widget.parent(),type(self.w.gcode_editor.editor)):
        print('G-code Editor')
    elif isinstance(widget,type(self.w.gcodegraphics)):
        print('G-code Display')
    elif isinstance(widget.parent(),type(self.w.mdihistory) ):
        print('MDI History')
```

Beachten Sie, dass wir manchmal mit `widget`, manchmal mit `widget.parent()` vergleichen.

Das liegt daran, dass *einige QtVCP-Widgets aus mehreren **Sub-Widgets*** aufgebaut sind und letztere tatsächlich den Fokus erhalten; daher müssen wir **den Elternteil** dieser Sub-Widgets überprüfen.

In anderen Fällen ist das Hauptwidget das, was den Fokus erhält, z. B. das G-Code-Anzeige-Widget kann so eingestellt werden, dass es den Fokus annimmt. In diesem Fall gibt es keine Unter-Widgets darin, so dass ein Vergleich mit `widget.parent()` den Container ergeben würde, der das G-Code-Widget enthält.

12.12.17. Lesen Sie die Optionen für die Ladezeit der Befehlszeile

Einige Panels benötigen zum Zeitpunkt des Ladens Informationen zur Einrichtung/Optionen. QtVCP deckt diese Anforderung mit *-o*-Optionen ab.
Das *-o*-Argument eignet sich für einige wenige, relativ kurze Optionen, die der Ladebefehlszeile hinzugefügt werden können.
Für umfangreichere Informationen ist das Lesen einer INI- oder Einstellungsdatei wahrscheinlich eine bessere Idee.

In der Befehlszeile können mehrere *-o*-Optionen verwendet werden, so dass Sie sie entschlüsseln müssen.
self.w.USEROPTIONS_' enthält alle gefundenen *-o*-Optionen als eine Liste von Strings. Sie müssen parsen und definieren, was akzeptiert wird und was damit zu tun ist.

Beispielcode zum Abrufen der -o Optionen für Kameranummer und Fenstergröße

```
    def initialized__(self):

        # setzt eine Standard-Kameranummer
        number = 0

        # prüfen, ob es überhaupt -o Optionen gibt
        if self.w.USEROPTIONS_ is not None:

            # wenn im Debug-Modus die Optionen auf dem Terminal ausgegeben werden
            LOG.debug('cam_align user options: {}'.format(self.w.USEROPTIONS_))

            # Gehe die gefundenen Optionen eine nach der anderen durch
            for num, i in enumerate(self.w.USEROPTIONS_):

                # wenn die Option -o 'size=' enthält, wird angenommen, dass es die Breite
und Höhe des Fensters ist
```

```
                # Überschreibe die Standardbreite und -höhe des Fensters
                if 'size=' in self.w.USEROPTIONS_[num]:
                    try:
                        strg = self.w.USEROPTIONS_[num].strip('size=')
                        arg = strg.split(',')
                        self.w.resize(int(arg[0]),int(arg[1]))
                    except Exception as e:
                        print('Error with cam_align size setting:',self.w.USEROPTIONS_
[num])

                #  # wenn die Option -o 'camnumber=' enthält, wird angenommen, dass dies
die zu verwendende Kameranummer ist
                elif 'camnumber=' in self.w.USEROPTIONS_[num]:
                    try:
                        number = int(self.w.USEROPTIONS_[num].strip('camnumber='))
                    except Exception as e:
                        print('Fehler mit cam_align Kamera Auswahl - das ist keine Nummer
- verwende 0')

        # setzt die Kameranummer entweder als Standard oder wenn die Option -o die
Variable 'number' geändert hat, auf diese Nummer.
        self.w.camview._camNum = number
```

12.12.18. G-Code zum Lesen von Qt Einstellungen

So erstellen Sie ein O-Word-Programm, um einen Eintrag aus einer QtDragon-Einstellungsdatei auszulesen und als G-Code-Parameter hinzuzufügen.
Beim Aufruf dieses O-Word wird der Parameter *toolToLoad* aktualisiert.
Dabei wird ein *Python Hot Comment* verwendet, um mit der eingebetteten Python-Instanz zu kommunizieren.
Eine Beschreibung finden Sie im Abschnitt „Remap“ der Dokumentation.

```
(filename myofile.ngc)
o<myofile> sub

;py,from interpreter import *
;py,import os
;py,from qtvcp.lib.preferences import Access

; Finde und gebe aus den Pfad für die Einstellungs-Datei
;py,CONFPATH = os.environ.get('CONFIG_DIR', '/dev/null')
; Nimm eine Anpassung für den Namen Deiner Einstellungsdatei vor
;py,PREFFILE = os.path.join(CONFPATH,'qtdragon.pref')
;py,print(PREFFILE)

; Erhalte Instanz für Einstellungen
;py,Pref = Access(PREFFILE)

; Lade eine Einstellung und gebe sie aus
;py,this.params['toolToLoad']=Pref.getpref('Tool to load', 0, int,'CUSTOM_FORM_ENTRIES')
;py,print('Tool to load->:',this.params['toolToLoad'])

; Liefere den Wert zurück
o<myofile> endsub [#<toolToLoad>]
```

```
M2
```

12.13. QtVCP-Entwicklung

12.13.1. Übersicht

Die Absicht von QtVCP ist es, **eine Infrastruktur zur Unterstützung von Bildschirmen und VCP-Panels für LinuxCNC** bereitzustellen.

Durch die Bereitstellung eines *vielfältigen Widgetsatzes* und die Unterstützung von *custom coding* hofft QtVCP, dass die Entwicklungsenergie in *einem Toolkit* verbraucht wird und nicht in ständiger Neuerfindung.

Durch die Verwendung desselben Toolkits für viele Bildschirme/Panels sollte es für die Benutzer einfacher sein, diese anzupassen/zu erstellen, und für die Entwickler sollte es einfacher sein, bei der Fehlersuche mit weniger Aufwand zu helfen.

QtVCP verwendet eine **von Qt Designer erstellte `.ui`-Datei** und eine **Python-Handler-Datei**

- zum **Laden und Steuern eines Bildschirms/Panels, das Qt-Widgets anzeigt** und
- zur **Steuerung von LinuxCNC's Motion Controller oder HAL Pins**.

Es gibt *vorgefertigte Bildschirme und Paneele*, die leicht von einem Benutzer geladen werden können, oder Benutzer können ihre eigenen erstellen/ändern.

QtVCP verwendet **Bibliotheken und benutzerdefinierte Widgets**, um einen Teil der Komplexität der Schnittstelle zu LinuxCNC zu verbergen. Durch die Verwendung der QtVCP-Bibliothek anstelle der LinuxCNC-Bibliothek können wir kleinere Änderungen am LinuxCNC-Code abmildern.

12.13.2. Integrierte Suchpfade

Integrierte Bildschirme und Bedienfelder werden in separaten Ordnern gespeichert:

- *Bildschirme* in *share/qtvcp/screens*
- *Panels* in `share/qtvcp/panels`
- *Stock Bilder* in `share/qtvcp/images `

Bildschirme und Bedienfelder werden nach ihrem Ordnernamen sortiert, der auch der Name ist, der zum Laden verwendet wird.

Im Ordner wäre:

- die `.ui` *Datei*,
- die *Handler-Datei*, und
- möglicherweise die `.qss` *Themen-Datei*.

12.13.3. QtVCP vom Startup bis zum Herunterfahren

QtVCP source befindet sich im Ordner *src/emc/usr_intf/qtvcp* des LinuxCNC-Quellcode Verzeichnisbaums.

QtVCP Startup

Beim ersten Start von QtVCP:

1. Es muss entscheiden, ob dieses Objekt ein Bildschirm oder ein Bedienelement (engl. panel) ist.
2. Es sucht und sammelt Informationen über Pfade von benötigten Dateien und nützlichen Ordnern.
3. Dann wird es:
   a. Erstellt die HAL-Komponente,
   b. Die Fenster-Instanz laden,
   c. Fügt Handler-Erweiterungen hinzu,
   d. einen Ereignisfilter installieren.

Jetzt werden die Fenster/Widgets instanziiert, die HAL-Pins werden aufgebaut. Dies initiiert auch die `+_init_hal()+` Funktion der Widgets. . Die Handler-Funktion `+initialized__()+` wird aufgerufen . Die `STATUS`-Bibliothek wird zur Aktualisierung gezwungen. . HAL Komponente ist zu diesem Zeitpunkt bereit. . Es werden eine Reihe von optionalen Schalterargumenten gesetzt, einschließlich des Aufrufs einer `POSTGUI`-HAL-Datei (falls es ein Bildschirm ist). . Terminate-Signale werden abgefangen und QtVCP fragt jetzt nach Ereignissen.

QtVCP Herunterfahren (engl. shutdown)

Wenn QtVCP schließlich zum Herunterfahren aufgefordert wird:

1. Sie ruft die Abschaltfunktionen in der Handler-Datei auf,
2. Die ‘STATUS’-Überwachung wird abgeschaltet
3. HAL-Komponenten wird terminiert (engl. killed)

12.13.4. Pfad-Informationen

Wenn QtVCP geladen wird, sammelt es Pfadinformationen.

Dieser ist in der Funktion `+__init__()+` der Handler-Datei als **`path`** (engl. für Pfad) verfügbar:

`IMAGEDIR`

Pfad zu mitgelieferten Bildern

`SCREENDIR`

Pfad der integrierten Motion-Controller-Bildschirme

PANELDIR

Pfad der eingebauten Zubehörtafeln

WORKINGDIR

Pfad, von dem aus QtVCP gestartet wurde

CONFIGPATH

Pfad der gestarteten Konfiguration

BASEDIR

Allgemeiner Pfad, der zur Ableitung aller Pfade verwendet wird

BASENAME

Generischer Name, der zur Ableitung aller Pfade verwendet wird

LIBDIR

Pfad der Python-Bibliothek von QtVCP

HANDLER

Pfad der Handler-Datei

XML

Pfad der .ui-Datei

DOMAIN

Pfad der Übersetzung

IS_SCREEN

Bildschirm-/Bedienfeldschalter

12.13.5. Eigenheiten

Diese versuchen, nicht offensichtliche Situationen abzudecken.

Sammlung von Fehlern

Die Fehlercodeerfassung von LinuxCNC kann nur von einem Ort aus gelesen werden.

Wenn es gelesen wird, ist es ***konsumiert***, d.h. *kein anderes Objekt kann es lesen*.

In QtVCP-Bildschirmen wird empfohlen, *das _Widget `ScreenOptions` zu verwenden, um das Lesen von Fehlern einzurichten*.

Fehler werden dann über **STATUS** *Signale* **an andere Objekte gesendet**.

Jogging-Rate

LinuxCNC hat keine interne Aufzeichnung der Jogging-Geschwindigkeit: *Sie müssen sie bei Beginn*

des Joggens festlegen.

QtVCP verwendet die `STATUS`-Bibliothek, um *die neuesten linearen und winkligen Jog-Raten zu verfolgen.*

Sie wird **immer in Maschineneinheiten pro Minute angegeben** und *muss umgerechnet werden, wenn Sie sich im Modus für Nicht-Maschineneinheiten befinden.*
Wenn Ihre Maschine also auf dem zölligen System basiert, Sie sich aber im metrischen Modus befinden, müssen Änderungen der Schrittgeschwindigkeit, die an die Funktionen "AKTION" gesendet werden, in zöllige Einheiten umgewandelt werden.
Wenn die Maschine auf metrischen Einheiten basiert und Sie sich im imperialen Modus befinden, müssen Änderungen der Tippgeschwindigkeit an die "AKTION"-Funktionen in metrischen Einheiten gesendet werden.
Für die Winkelgeschwindigkeit ändern sich die Einheiten im metrischen/imperialen Modus nicht, so dass Sie sie ohne Umrechnung an die Funktionen "AKTION" senden können.

Während es Ihnen freisteht, diesen Jogging-Datensatz beim Erstellen von Bildschirmen zu ignorieren, würde jeder, der Ihren Bildschirm modifiziert und die eingebauten Jog-Rate-Widgets verwendet, nicht die gewünschten Ergebnisse erhalten, da die **`DO_JOG`**-Funktion der `ACTION`-Bibliothek ihre Jog-Rate von der `STATUS`-Bibliothek erhält.

Tastenbelegung

WARNING	Die Zuordnung (engl. binding) von Tasten ist immer eine *schwierige Angelegenheit, die man nicht in allen Fällen richtig hinbekommt.*

Benutzerdefinierte Tastenbindungsfunktionen müssen *in der Handler-Datei* definiert werden.

Vor allem Widgets, die eine reguläre Tasteneingabe und kein Rütteln erfordern, sollten in der Funktion `processed_key_event__` überprüft werden.

Einstellungsdatei

Einige QtVCP Widgets verwenden die Präferenzdatei, um wichtige Informationen aufzunehmen.

Dies *erfordert, dass diese Einstellungsdatei früh* im Initialisierungsprozess des Widgets eingerichtet wird.
Der einfachste Weg, dies zu tun, ist die **Verwendung des `ScreenOptions`-Widgets**.

Spezielle Widget-Einrichtungsfunktionen

QtVCP sucht und ruft die Funktion `+_hal_init()+` auf, *wenn das Widget zum ersten Mal geladen wird.*

Diese wird nicht aufgerufen, wenn der Qt Designer Editor verwendet wird.

Nachdem diese Funktion aufgerufen wurde, hat das Widget Zugriff auf einige spezielle Variablen:

`self.HAL_GCOMP`

Die *HAL-Komponenten*-Instanz

`self.HAL_NAME`

Der Name dieses *Widgets* als String

`self.QT_OBJECT_`

dieses *Widgets PyQt-Objekt als Instanz*

`self.QTVCP_INSTANCE_`

Die *übergeordnete Ebene* des Bildschirms

`self.PATHS_`

Die _Instanz der Pfadbibliothek von QtVCP

`self.PREFS_`

Die *Instanz einer optionalen Präferenzdatei*

`self.SETTINGS_`

Das `Qsettings` *Objekt*

Wenn Sie ein benutzerdefiniertes Widget erstellen, _importieren Sie die Klasse **`+_HalWidgetBase+`** und spezialisieren Sie sie für dieses Verhalten durch Anlegen einer Unterklasse.

Dialoge

Dialoge (auch bekannt als "Pop-up-Fenster") werden *am besten mit dem* `ScreenOptions` *_widget* geladen, aber sie können auch im Qt Designer auf dem Bildschirm platziert werden.

Es spielt keine Rolle, wo auf dem Layout, aber *um sie auszublenden*, schalten Sie die Eigenschaft `state` auf true und dann auf false.

Wenn eine Einstellungsdatei vorhanden ist, merken sich die Dialoge standardmäßig ihre letzte Größe/Position.
Es ist möglich, dies außer Kraft zu setzen, so dass sie jedes Mal an der gleichen Stelle geöffnet werden.

Stile (Designs, engl. themes)

Es ist zwar möglich, Stile *im Qt Designer* zu setzen, aber es ist bequemer, sie später zu ändern, wenn sie alle in einer ***separaten*** **`.qss`** ***Datei*** gesetzt werden. Diese Datei sollte am *gleichen Ort wie die Handler-Datei* abgelegt werden.

Chapter 13. Programmierung der Benutzeroberfläche

13.1. Panelui

13.1.1. Einführung

Panelui ist eine Nicht-Echtzeit-Komponente zur Anbindung von Schaltflächen an LinuxCNC oder HAL:

- Es dekodiert MESA 7I73-artige Key-Scan-Codes und ruft die entsprechende Routine auf.
- Die Eingabe erfolgt über eine Echtzeitkomponente - Sampler. Der Sampler erhält seine Eingaben entweder von der Komponente MESA 7I73 oder sim_matrix_kb.
- Panelui kann mithilfe einer INI-Textdatei konfiguriert werden, um Schaltflächentypen, HAL-Pin-Typen und/oder Befehle zu definieren.
- Sie kann mit einer Python-basierten "Handler"-Datei um Funktionen erweitert werden.

Während die eigentlichen Eingabetasten träge sein müssen, verwendet Panelui diese Eingabe für die Ausgabe von Toggle-, Radio- oder Tasterschaltungen.

13.1.2. Laden von Befehlen

Der Befehl zum Laden von panelui (mit optionalem Schalter -d debug):

```
loadusr -W panelui -d
```

Dadurch wird panelui initialisiert, das im Konfigurations- oder Benutzerordner nach der INI-Datei panelui.ini sucht.

Mit diesem Befehl kann man die INI-Datei validieren:

```
loadusr pyui
```

Damit wird die Datei panelui.ini gelesen, versucht zu korrigieren und dann gespeichert. Eventuelle Fehler werden diese auf dem Terminal ausgegeben.

Einer typischen HAL-Datei werden diese Befehle hinzugefügt:

```
# Befehle, die für das Laden von Panelui benötigt werden
#
# sampler wird für panelui benötigt
# cfg= muss für panelui immer u sein. depth legt den verfügbaren Puffer fest
loadrt sampler cfg=u depth=1025

#unkommentiert, um die panelui INI-Datei zu validieren
#loadusr pyui

# -d = Fehlersuche, -v = ausführliche Fehlersuche
# -d zeigt Ihnen die Tastenkennzeichnung und die aufgerufenen Befehle
```

```
# -v ist für Informationen über den Entwickler
loadusr -W panelui -d

# mit simulierten Tasten anstelle der MESA 7I73-Karte
# also laden wir die Komponente sim_matrix_kb, um die HAL-Pins in Keyscan-Codes
umzuwandeln
loadrt sim_matrix_kb

# Verbinden Sie die Komponenten miteinander.
# sampler spricht intern mit panelui
net key-scan sim-matrix-kb.0.out
net key-scan sampler.0.pin.0

# panelui Komponenten zu einem Thread hinzufügen

addf sim-matrix-kb.0    servo-thread
addf sampler.0          servo-thread
```

13.1.3. panelui.ini Dateireferenz

Schlüsselwörter *(engl. keywords)*

- KEY= Hier wird die Taste angegeben, auf die der Button reagiert. Es kann KEINE oder eine Zeilennummer und eine Spaltennummer sein, z. B. R1C2. Eine Zeile und Spalte kann nur einmal verwendet werden.
- OUTPUT= This sets the Button's output type, eg SINT, UINT, REAL, BOOL, NONE, COMMAND, ZMQ.
- DEFAULT= Hiermit wird die Startausgabe der Gruppe oder der Schaltfläche festgelegt.
- GROUP= Bezeichnet bei Radiobuttons die Gruppe, mit innerhalb welcher der Button interagiert.
- GROUP_OUTPUT= legt den Ausgang fest, den der Gruppenpin hat, wenn dieser Button aktiv ist.
- STATUS_PIN= Wenn TRUE, wird ein HAL-Pin hinzugefügt, der den aktuellen Zustand des Button wiedergibt.
- TRUE_STATE= legt den Ausgang fest, den der HAL-Pin hat, wenn der Button TRUE ist.
- FALSE_STATE= legt den OUTPUT fest, den der HAL-Pin erhält, wenn die Taste FALSE ist.
- TRUE_COMMAND= legt den Befehl und die Argumente fest, die aufgerufen werden, wenn der Button TRUE ist.
- FALSE_COMMAND= legt den Befehl und die Argumente fest, die aufgerufen werden, wenn der Button FALSE ist.
- TRUE_FUNCTION= legt die ZMQ-Nachrichtenfunktion und Argumente fest, die aufgerufen werden sollen, wenn der Button TRUE ist.
- FALSE_FUNCTION= legt die ZMQ-Nachrichtenfunktion und die Argumente fest, die aufgerufen werden sollen, wenn der Button FALSE ist.

HAL Prefix

```
[HAL_PREFIX]
    NAME= IhrName
```

Damit kann man den Präfix der HAL-Pins von *panelui* auf einen beliebigen Namen ändern.

ZMQ Messaging-Einrichtung

```
[ZMQ_SETUP]
  TOPIC = 'QTVCP'
  SOCKET = 'tcp://127.0.0.1:5690'
  ENABLE = True
```

Damit wird das ZMQ-basierte Messaging eingerichtet und aktiviert. TOPIC und SOCKET müssen mit dem empfangenden Programm übereinstimmen.

Radio Buttons

Bei Radiobutons kann jeweils nur eine Taste in der Gruppe aktiv sein. Jede Gruppe hat ihren eigenen Ausgangspin, der von jeder Taste in der Gruppe getrennt ist. Radiobutton-Definitionen beginnen mit dem Text "RADIO_BUTTON" in einfachen Klammern.

```
[RADIO_BUTTONS]
  # The double bracket section(s) define the group(s) of radio buttons.
  # The group name must be unique and is case sensitive.
  # Groups output is controlled by what button is active not directly by keycode.
  # DEFAULT references a button in the group by name and is case sensitive.
  [[group1_name]]
    KEY = NONE
    OUTPUT = REAL
    DEFAULT = small
    # The triple bracket sections define the buttons in this group.
    # button names must be unique and are case sensitive.
    # There must be at least two buttons in a group.
    #
    # This button, named 'small'is controller by the row 0 column 1 key.
    # It will cause the group output to be .0001 when it is pressed.
    # It has no output of its own, but has a status
    # pin which will follow its current state.
    # since this button is in a group, DEFAULT has no bearing.
    # since OUTPUT in not 'COMMAND' _COMMAND entries are ignored.
    [[[small]]]
      KEY = R0C1
      GROUP = group1_name
      GROUP_OUTPUT = .0001
      OUTPUT = NONE
      STATUS_PIN = True
      TRUE_STATE = TRUE
      FALSE_STATE = FALSE
      TRUE_COMMAND = NONE, NONE
      FALSE_COMMAND = NONE, NONE
      DEFAULT = false
    # This button, named 'large' is controller by the row 0 column 2 key.
    # It will cause the group output to be 1000 when it is pressed.
    # It has a SINT output of its own, will be 20 on true and 0 on false.
    # It also has a status pin which will follow its current state.
    # since this button is in a group, DEFAULT has no bearing.
    # since OUTPUT in not 'COMMAND' _COMMAND entries are ignored.
    [[[large]]]
```

```
      KEY = R0C2
      GROUP = group1_name
      GROUP_OUTPUT = 1000
      OUTPUT = SINT
      STATUS_PIN = True
      TRUE_STATE = 20
      TRUE_COMMAND = NONE, NONE
      FALSE_COMMAND = NONE, NONE
      FALSE_STATE = 0
      DEFAULT = false
```

Wechsel-Buttons *(engl. toggle buttons)*

Togglebuttons ändern ihren Zustand nur bei jedem Drücken der Taste. Toggle-Button-Definitionen beginnen mit dem Text "TOGGLE_BUTTON" in einfachen Klammern.

```
[TOGGLE_BUTTONS]
  # Each button name inside double brackets, must be unique and is case sensitive.
  # This button, named 'tool_change'is controller by the row 2 column 5 key.
  # It has a BOOL output, will output 1 on true state and 0 on false state.
  # It also has a status pin which will follow its current state.
  # DEFAULT sets this to true when first initialized.
  # The _COMMAND are not used since OUTPUT is not set to COMMAND but validation will
  # add the lines regardless
  [[tool_change]]
    KEY = R2C5
    OUTPUT = BOOL
    TRUE_COMMAND = NONE, NONE
    FALSE_COMMAND = NONE, NONE
    STATUS_PIN = True
    DEFAULT = TRUE
    TRUE_STATE = 1
    FALSE_STATE = 0
```

Momentary Buttons

Momentane Buttons sind wahr, wenn sie gedrückt werden, und falsch, wenn sie losgelassen werden. Button-Definitionen beginnen mit dem Text "MOMENTARY_BUTTON" in einfachen Klammern.

```
[MOMENTARY_BUTTONS]
  # Jeder Tastenname in doppelten Klammern muss eindeutig sein und Groß- und
Kleinschreibung wird unterschieden.
  # Diese Taste mit dem Namen 'spindle_rev' wird von der Taste in Zeile 2 Spalte 3
gesteuert.
  # Sie hat einen COMMAND-Ausgang, verwendet also TRUE_COMMAND und FALSE_COMMAND.
  # Er hat auch einen Status-Pin, der seinen aktuellen Zustand anzeigt.
  # COMMANDs haben einen Befehlsnamen und dann alle erforderlichen Argumente.
  # Dieser TRUE_COMMAND ruft einen internen Befehl zum Starten der Spindel im
Rückwärtsgang mit 200 U/min auf.
  # Wenn die Spindel bereits gestartet ist, wird die Drehzahl erhöht.
  # DEFAULT wird nicht mit Momentary-Buttons verwendet.
  # Die _STATE werden nicht verwendet, da OUTPUT auf COMMAND gesetzt ist, aber die
Validierung wird
  # die Zeilen dennoch hinzufügen.
  [[spindel_drehzahl]]
```

```
    KEY = R2C3
    OUTPUT = COMMAND
    TRUE_COMMAND = SPINDLE_REVERSE_INCREASE, 200
    FALSE_COMMAND = None, NONE
    STATUS_PIN = True
    DEFAULT = FALSE
    TRUE_STATE = 1
    FALSE_STATE = 0
```

13.1.4. Übersicht zu Internen Anweisungen

Es gibt eine Reihe von internen Befehlen, die Sie verwenden können.

home_selected

- erforderliches Argument: Achsennummer (int)

unhome_selected

- erforderliches Argument: Achsennummer (int)

spindle_forward_adjust

- optionales Argument: Anfangsdrehzahl (int) - Standardwert 100
- Beschreibung: Wenn die Spindel angehalten ist, startet sie in Vorwärtsrichtung. Wenn sie bereits läuft, erhöht oder verringert sie die Drehzahl, je nachdem, in welche Richtung die Spindel läuft.

spindle_forward

- optionales Argument: Anfangsdrehzahl (int) - Standardwert 100

spindle_reverse

- optionales Argument: Anfangsdrehzahl (int) - Standardwert 100

spindle_reverse_adjust

- optionales Argument: Anfangsdrehzahl (int) - Standardwert 100
- Beschreibung: Wenn die Spindel angehalten wird, startet sie in umgekehrter Richtung. Wenn sie bereits läuft, erhöht oder verringert sie die Drehzahl, je nachdem, in welche Richtung die Spindel läuft.

spindle_faster

- Beschreibung: erhöht die Spindeldrehzahl um 100 RPM

spindle_slower

- Beschreibung: Verringert die Spindeldrehzahl um 100 RPM, bis die Drehzahl 100 beträgt.

set_linear_jog_velocity

- erforderliches Argument: Geschwindigkeit in Zoll pro Minute (Float)
- Beschreibung: setzt die Jog-Geschwindigkeit auf den Achsen 0,1,2,6,7,8 (X,Y,Z,U,V,W)

set_angular_jog_velocity

- erforderliches Argument: Geschwindigkeit in Grad pro Minute (Float)
- Beschreibung: Setzt die Jog-Geschwindigkeit auf Achse 3,4,5 (A.B.C)

continuous_jog

- erforderliche Argumente: Achsennummer (int), Richtung (int)

incremental_jog

- erforderliche Argumente: Achsennummer (int), Richtung (int), Abstand (float)

quill_up

- optionale Argumente: absolute Position der Z-Achse der Maschine (Float)
- Beschreibung: Z-Achse auf die angegebene Maschinenposition fahren

feed_hold

- Erforderliches Argument: Zustand (bool 0 oder 1)

feed_override

- erforderliches Argument: Rate (float)

rapid_override

- erforderliches Argument: Rate (float 0-1)

spindle_override

- erforderliches Argument: Rate (float)

max_velocity

- erforderliches Argument: Rate (float)

optional_stop

- Erforderliches Argument: Zustand (bool 0 oder 1)

block_delete *(engl. für Block löschen)*

- Erforderliches Argument: Zustand (bool 0 oder 1)

single_block

- Erforderliches Argument: Zustand (bool 0 oder 1)

smart_cycle_start

- Beschreibung: Wenn im Leerlauf, startet G-Code-Programm, wenn angehalten wird eine Zeile ausgeführt.

re_start line

- erforderliches Argument: Zeilennummer (int)

mdi_and_return

- erforderliches Argument: G-Code-Befehl(e)

- Beschreibung: Zeichnet den aktuellen Modus auf, ruft Befehle auf und kehrt dann zum Modus zurück.

mdi

- erforderliches Argument: G-Code-Befehl(e)
- Beschreibung: Setzt den Modus auf MDI, ruft Befehle auf.

13.1.5. ZMQ-Nachrichten

Panelui kann ZMQ-basierte Nachrichten beim Drücken von Tasten senden.
Auf diese Weise kann panelui mit anderen Programmen wie QtVCP-Bildschirmen interagieren.

```
[TOGGLE_BUTTONS]
  [zmq_test]
    KEY = R2C3
    OUTPUT = ZMQ
    TRUE_FUNCTION = ZMQ_BUTTON, 200
    FALSE_FUNCTION = ZMQ_BUTTON, 0
    STATUS_PIN = False
    DEFAULT = FALSE
    TRUE_STATE = 1
    FALSE_STATE = 0
```

Hier ist ein Beispielprogramm, das die Nachricht empfängt und auf dem Terminal ausgibt.

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# Definition des Sockets mit Hilfe des "Context".
sock = context.socket(zmq.SUB)

# Definieren des Abonnements und der zu akzeptierenden Nachrichten ohne Einschränkung der
Themen.
topic = "" # alle Themen
sock.setsockopt(zmq.SUBSCRIBE, topic)
sock.connect("tcp://127.0.0.1:5690")

while True:
    topic, message = sock.recv_multipart()
    print('{} sent message:{}'.format(topic,json.loads(message)))
```

13.1.6. Handler Dateierweiterung

Eine spezielle Datei kann verwendet werden, um benutzerdefinierten Python-Code hinzuzufügen, der als Befehle verfügbar ist. panelui_handler.py muss in Python geschrieben und im Konfigurationsordner abgelegt werden. Wenn panelui dort eine Datei findet, fügt es deren Funktionsaufrufe zu den verfügbaren Befehlen hinzu. Hier ist ein Beispiel für eine Handler-Datei, die zwei Funktionen hinzufügt

- hello_world und cycle_mode:

```
# standard handler call - This will always be required
def get_handlers(linuxcnc_stat, linucnc_cmd, commands, master):
     return [HandlerClass(linuxcnc_stat, linucnc_cmd, commands, master)]

# Ebenfalls erforderlich - Handler-Klassenklasse HandlerClass:

    # Dies wird ein ziemlicher Standard sein, um Zugriff auf alles zu erhalten.
    # linuxcnc_stat: ist die Python-Status-Instanz von LinuxCNC
    # linuxcnc_cmd: ist die Python-Befehlsinstanz von LinuxCNC
    # commands: ist die Befehlsinstanz, damit man die internen Routinen aufrufen kann
    # master: ermöglicht den Zugriff auf die Master-Funktionen/Daten

    def __init__(self, linuxcnc_stat, linuxcnc_cmd, commands, master):
        self.parent = commands
        self.current_mode = 0

    # command functions are expected to have this layout:
    # def some_name(self, widget_instance, arguments from widget):
    # widget_instance gives access to the calling widget's function/data
    # arguments can be a list of arguments, a single argument, or None
    # depending on what was given in panelui's INI file.
    def hello_world(self, wname, m):
        # print to terminal so we know it worked
        print('\nHello world\n')
        print(m)     # print the argument(s)
        print(wname.metadata)    # Print the calling widgets internal metadata (from
config file)

        # Aufruf eines MDI-Befehls zum Drucken einer Nachricht in LinuxCNC.
        # Dies setzt voraus, dass LinuxCNC referenziert ist, aber das wird nicht
überprüft.
        # übergeordnete Befehle erwarten eine widget_instance - None wird ersetzt
        self.parent.mdi(None,'(MSG, Hallo Linuxcnc Welt!)')

    # Jeder Aufruf dieser Funktion schaltet den Modus von LinuxCNC um.
    def cycle_mode(self, wname, m):
        if self.current_mode == 0:
            self.current_mode = 1
            self.parent.set_mdi_mode()
        elif self.current_mode == 1:
            self.current_mode = 2
            self.parent.set_auto_mode()
        else:
            self.current_mode = 0
            self.parent.set_manual_mode()
        print(self.current_mode)

    # Boiler code, braucht man oft
    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

13.2. Das LinuxCNC Python-Modul

Diese Dokumentation beschreibt das `linuxcnc` Python-Modul, das eine Python-API für die Kommunikation mit LinuxCNC bereithält.

13.2.1. Einführung

Benutzeroberflächen steuern LinuxCNC-Aktivitäten durch Senden von NML-Nachrichten an die LinuxCNC-Task-Controller, und überwachen die Ergebnisse durch die Beobachtung der LinuxCNC-Status-Struktur, sowie des Fehlerberichterstattung Kanals.

Der programmatische Zugriff auf NML erfolgt über eine C++-API; die wichtigsten Teile der NML-Schnittstelle zu LinuxCNC sind jedoch auch für Python-Programme über das Modul `linuxcnc` verfügbar.

Neben der NML-Schnittstelle zu den Befehls-, Status- und Fehlerkanälen enthält das Modul `linuxcnc` auch:

- Unterstützung für das Lesen von Werten aus INI-Dateien

13.2.2. Verwendungsmuster für die LinuxCNC NML-Schnittstelle

Das allgemeine Muster für die Verwendung von `linuxcnc` ist in etwa wie folgt:

- Import des `linuxcnc` Moduls.
- Bei Bedarf Verbindungen zu den Befehls-, Status- und Fehler-NML-Kanälen herstellen.
- Den Statuskanal abfragen, entweder regelmäßig oder nach Bedarf.
- Vor dem Senden eines Befehls anhand des Status feststellen, ob dies tatsächlich zulässig ist (z. B. hat es keinen Sinn, einen *Ausführen* Befehl zu senden, wenn sich die Aufgabe im NOTAUS (engl. ESTOP)-Zustand befindet oder der Interpreter nicht im Leerlauf ist).
- Den Befehl mit einer der Methoden des Befehlskanals `linuxcnc` senden.

Um Nachrichten aus dem Fehlerkanal abzurufen, rufen Sie den Fehlerkanal regelmäßig ab und verarbeiten alle abgerufenen Nachrichten.

- Den Statuskanal abfragen, entweder regelmäßig oder nach Bedarf.
- Drucken Sie eine Fehlermeldung aus, und untersuchen Sie den Ausnahme (engl. exception)-Code.

Das Modul `linuxcnc` definiert auch den Python-Ausnahmetyp "error", um Fehlerberichte zu unterstützen.

13.2.3. Auslesen des LinuxCNC Status mit dem linuxcnc Python Modul

Hier ist ein Python-Fragment, um den Inhalt des Objekts `linuxcnc.stat` zu untersuchen, das mehr als 80 Werte enthält (führen Sie es aus, während LinuxCNC läuft, um typische Werte zu erhalten):

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
import sys
import linuxcnc
try:
    s = linuxcnc.stat() # erstellt Verbindung zu Status-Kanal
    s.poll() # erhält aktuelle Werte
except linuxcnc.error, detail:
    print("Fehler", detail)
    sys.exit(1)
for x in dir(s):
    if not x.startswith("_"):
        print(x, getattr(s,x))
```

Das linuxcnc Modul verwendet den standardmäßig einkompilierten Pfad zur NML-Konfigurationsdatei, sofern er nicht überschrieben wird, siehe Lesen von INI-Datei Werten für ein Beispiel.

linuxcnc.stat-Attribute

acceleration (engl. für Beschleunigung)

(returns float) - Standardbeschleunigung, spiegelt den INI-Eintrag [TRAJ]DEFAULT_ACCELERATION wider.

active_queue (engl. für aktive Queue)

(returns integer) - Anzahl der geplanten ineinander übergehenden Bewegungen.

actual_position (engl. für Ist-Position)

(gibt ein Tupel von Floats zurück) - aktuelle Position der Flugbahn (x y z a b c u v w) in Maschineneinheiten.

adaptive_feed_enabled (engl. für Adaptiver Vorschub aktiviert)

(returns boolean) - Status der adaptiven Vorschubüberschreibung (0/1).

ain

(gibt ein Tupel von Floats zurück) - aktueller Wert der analogen Eingangspins.

actual_position (engl. für *Winkeleinheiten)

(gibt Float zurück) - Maschinenwinkeleinheiten pro Grad, entspricht dem [TRAJ]ANGULAR_UNITS INI Eintrag.

aout

(gibt ein Tupel von Floats zurück) - aktueller Wert der analogen Ausgangspins.

axes (*Entfernt mit Version 2.9*)

verwenden Sie stattdessen `axis_mask.bit_count()`, um die Anzahl der konfigurierten Achsen zu erhalten.

axis (engl. für Achse)

(gibt ein Tupel von Dicts zurück) - spiegelt die aktuellen Achsenwerte wider. Siehe Das Achsen-Wörterbuch.

axis_mask (engl. für Achsen-Maske)

(gibt ganze Zahl zurück) - Maske der verfügbaren Achsen, wie durch `[TRAJ]COORDINATES` in der INI-Datei definiert. Gibt die Summe der Achsen X=1, Y=2, Z=4, A=8, B=16, C=32, U=64, V=128, W=256 zurück.

block_delete (engl. für Block löschen)

(returns boolean) - Status des Flags "Block löschen".

call_level (engl. für Aufrufebene)

(gibt ganze Zahl zurück)` - aktuelle Tiefe des Unterprogramms. - 0 wenn nicht in einem Unterprogramm, (Verschachterlungs-/Rekursions-)tiefe wenn nicht anders angegeben.

command (engl. für Befehl)

(returns string) - aktuell ausgeführter Befehl.

current_line (engl. für aktuelle Zeile)

(returns integer) - aktuell ausgeführte Zeile.

current_vel (engl. für aktuelle Geschwindigkeit)

(returns float) - aktuelle Geschwindigkeit in Benutzereinheiten pro Sekunde.

cycle_time (engl. für Zyklus-Zeit)

(returns float) - Thread-Periode

debug

(returns integer) - Debug-Flag aus der INI-Datei.

delay_left (engl. für verbleibende Verzögerung)

(returns float) - verbleibende Zeit des Verweilzeitbefehls (G4), Sekunden.

din

(gibt ein Tupel von Ganzzahlen zurück) - aktueller Wert der digitalen Eingangspins.

distance_to_go (engl. für verbleibende Entfernung)

(returns float) - verbleibende Entfernung der aktuellen Bewegung, wie vom Trajektorienplaner gemeldet.

dout

(gibt ein Tupel von Ganzzahlen zurück) - aktueller Wert der digitalen Ausgangspins.

dtg

(returns tuple of floats) - verbleibende Entfernung der aktuellen Bewegung für jede Achse, wie vom Trajektorienplaner gemeldet.

echo_serial_number

(returns integer) – Die Seriennummer des letzten abgeschlossenen Befehls, der von einer Benutzeroberfläche an die Aufgabe gesendet wurde. Alle Befehle tragen eine fortlaufende Nummer.

Sobald der Befehl ausgeführt wurde, wird seine Seriennummer in `echo_serial_number` widergespiegelt.

enabled (engl. für aktiviert)

(returns boolean) - Trajektorienplaner aktiviert Flag.

estop (engl. für Notaus)

(returns integer) - Gibt entweder STATE_ESTOP zurück oder nicht.

exec_state

(gibt ganze Zahl zurück) - Status der Aufgabenausführung. Einer von EXEC_ERROR, EXEC_DONE, EXEC_WAITING_FOR_MOTION, EXEC_WAITING_FOR_MOTION_QUEUE, EXEC_WAITING_FOR_IO, EXEC_WAITING_FOR_MOTION_AND_IO, EXEC_WAITING_FOR_DELAY, EXEC_WAITING_FOR_SYSTEM_CMD, EXEC_WAITING_FOR_SPINDLE_ORIENTED.

feed_hold_enabled

(returns boolean) - Flag für Feed-Hold aktivieren.

feed_override_enabled

(returns boolean) - Flag für Feed-Override aktivieren.

Vorschubrate

(returns float) - aktuelle Überschreibung der Vorschubrate, 1,0 = 100 %.

Datei

(returns string) - aktuell geladener G-Code-Dateiname mit Pfad.

Flut

(gibt ganze Zahl zurück) - Flutungsstatus, entweder FLOOD_OFF oder FLOOD_ON.

g5x_index

(gibt eine ganze Zahl zurück) - derzeit aktives Koordinatensystem, G54=1, G55=2 usw.

g5x_offset

(gibt ein Tupel von Floats zurück) - Offset des aktiven Koordinatensystems.

g68_active

(returns integer) - a tilted work plane (G68.2) is in effect.

g68_offset

(returns tuple of floats) - origin of the tilted work plane, in the coordinate system it was defined in.

g68_rotation

(returns tuple of floats) - rotation of the tilted work plane, nine values row by row; its columns are the plane's X, Y and Z.

g92_offset

(returns tuple of floats) - Pose des aktuellen g92-Offsets.

gcodes

(liefert ein Tupel von Ganzzahlen) – Aktive G-Codes für jede Modalgruppe.
Die Ganzzahlen entsprechen den nominalen G-Code-Nummern multipliziert mit 10. (Beispiele: 10 = *G1*, 430 = *G43*, 923 = *G92.3*)

heartbeat

(returns integer) - motion controller heartbeat counter. Increments every servo cycle. Rate is determined by `[EMCMOT]SERVO_PERIOD` in the INI file.

homed

(gibt ein Tupel von ganzen Zahlen zurück) - aktuell referenzierte Gelenke, mit 0 = nicht referenziert, 1 = referenziert.

id

(gibt ganze Zahl zurück) - aktuell ausgeführte Bewegungskennung (engl. motion ID).

ini_filename

(returns string) - Pfad zur INI-Datei, die an linuxcnc übergeben wird.

inpos

(gibt einen booleschen Wert zurück) - Maschine-in-Position-Flag.

input_timeout

(returns boolean) - Flag für M66-Timer läuft.

interp_state

(gibt ganze Zahl zurück) - aktueller Zustand des RS274NGC-Interpreters. Einer von INTERP_IDLE, INTERP_READING, INTERP_PAUSED, INTERP_WAITING.

interpreter_errcode

(gibt ganze Zahl zurück) - aktueller RS274NGC-Interpreter-Rückgabecode. Einer von INTERP_OK, INTERP_EXIT, INTERP_EXECUTE_FINISH, INTERP_ENDFILE, INTERP_FILE_NOT_OPEN, INTERP_ERROR. siehe src/emc/nml_intf/interp_return.hh

joint

(gibt ein Tupel von dicts zurück) - spiegelt die aktuellen Gelenkwerte wider. Siehe Das gemeinsame Wörterbuch.

joint_actual_position

(gibt ein Tupel von Floats zurück) - tatsächliche Gelenkpositionen.

joint_position

(gibt Tupel von Floats zurück) - Gewünschte gemeinsame Positionen.

joints

(returns integer) - Anzahl der Joints. Reflektiert [KINS]JOINTS INI-Wert.

kinematics_type

(returns integer) - Der Typ der Kinematik. Einer von:

- KINEMATICS_IDENTITY
- KINEMATIKEN_FORWARD_ONLY
- KINEMATICS_INVERSE_ONLY
- KINEMATICS_BOTH

limit

(gibt Tupel von ganzen Zahlen zurück) - Achsengrenzwertmasken. minHardLimit=1, maxHardLimit=2, minSoftLimit=4, maxSoftLimit=8.

linear_units

(returns float) - Maschine lineare Einheiten pro mm, spiegelt [TRAJ]LINEAR_UNITS INI-Wert wider.

max_acceleration

(returns float) - maximale Beschleunigung. Reflektiert [TRAJ]MAX_ACCELERATION.

max_velocity

(returns float) - maximale Geschwindigkeit. Gibt die aktuelle maximale Geschwindigkeit wieder. Wenn es nicht durch halui.max-velocity oder ähnliches modifiziert wird, sollte es [TRAJ]MAX_VELOCITY widerspiegeln.

mcodes

(gibt ein Tupel von 10 ganzen Zahlen zurück) - derzeit aktive M-Codes.

mist

(gibt ganze Zahl zurück) - Nebelzustand, entweder MIST_OFF oder MIST_ON

motion_line

(gibt die ganze Zahl zurück) - Die Quellzeilennummernbewegung wird derzeit ausgeführt. Zusammenhang mit *id* unklar.

motion_mode

(returns integer) - Dies ist der Modus des Motion Controllers. Einer von TRAJ_MODE_COORD, TRAJ_MODE_FREE, TRAJ_MODE_TELEOP.

motion_type

(returns integer) - Der Typ der aktuell ausgeführten Bewegung. Einer von:

- MOTION_TYPE_TRAVERSE
- MOTION_TYPE_FEED
- MOTION_TYPE_ARC
- MOTION_TYPE_TOOLCHANGE
- MOTION_TYPE_PROBING

- MOTION_TYPE_INDEXROTARY
- Oder 0, wenn gerade keine Bewegung stattfindet.

optional_stop

(gibt die ganze Zahl zurück) - Option Stopp-Flag.

paused

(gibt einen booleschen Wert zurück) - *Bewegung pausiert*-Flag.

single_stepping

(returns boolean) - `motion single_stepping` flag.

pocket_prepped

(gibt eine ganze Zahl zurück) - Ein Tx-Befehl wurde ausgeführt, und diese Tasche ist vorbereitet. -1 wenn keine vorbereitete Tasche.

poll()

-(eingebaute Funktion) Methode zur Aktualisierung der aktuellen Statusattribute.

position

(gibt das Tupel von Floats zurück) - Trajektorienposition.

probe_tripped

(returns boolean) - Flag, wahr, wenn die Sonde ausgelöst hat (Latch).

probe_val

(returns integer) - spiegelt den Wert des Pins *motion.probe-input* wider.

probed_position

(gibt ein Tupel von Floats zurück) - Position, an der die Sonde ausgelöst wurde.

probiert

(returns boolean) - flag, true, wenn ein Prüfpunktvorgang ausgeführt wird.

program_units

(gibt ganze Zahl zurück) - eine von CANON_UNITS_INCHES=1, CANON_UNITS_MM=2, CANON_UNITS_CM=3

queue

(gibt ganze Zahl zurück) - aktuelle Größe der Warteschlange des Trajektorienplaners.

queue_full

(returns boolean) - Die Trajektorienplaner-Warteschlange ist voll.

rapidrate

(returns float) - Eilgang Übersteuerungs-Faktor.

read_line

(returns integer) - Zeile, die der RS274NGC-Interpreter gerade liest.

rotation_xy

(returns float) - aktueller XY-Rotationswinkel um die Z-Achse.

settings

(gibt ein Tupel von Floats zurück) - aktuelle Interpretereinstellungen.
settings[0] = Sequenznummer,
settings[1] = Vorschubgeschwindigkeit,
settings[2] = Geschwindigkeit,
settings[3] = `G64 P` Blendtoleranz,
settings[4] = `G64 Q` naive CAM Toleranz.

spindle

' (gibt Tupel von Dicts zurück) ' - gibt den aktuellen Spindelstatus zurück, siehe das Spindel-Wörterbuch.

spindles

(returns integer) - Anzahl der Spindeln. Reflektiert den INI-Wert von `[TRAJ]SPINDLES`.

state

(returns integer) - aktueller Status der Befehlsausführung. Einer von `RCS_DONE`, `RCS_EXEC`, `RCS_ERROR`.

task_mode

(returns integer) - aktueller Aufgabenmodus. Einer von `MODE_MDI`, `MODE_AUTO`, `MODE_MANUAL`.

task_paused

(gibt die ganze Zahl zurück) - Flag "Aufgabe angehalten".

task_state

(returns integer) - aktueller Aufgabenzustand. Einer von `STATE_ESTOP`, `STATE_ESTOP_RESET`, `STATE_ON`, `STATE_OFF`.

tool_in_spindle

(gibt ganze Zahl zurück) - aktuelle Werkzeugnummer.

taskbeat

(returns integer) - task main loop heartbeat counter. Increments every task cycle. Rate is determined by `[TASK]CYCLE_TIME` in the INI file.

tool_from_pocket

(liefert eine ganze Zahl) - Platznummer für das aktuell geladene Werkzeug (0, wenn kein Werkzeug geladen ist).

tool_offset

(returns tuple of floats) - Versatzwerte des aktuellen Werkzeugs.

tool_table

(gibt ein Tupel von tool_results zurück) - Liste der Werkzeugeinträge. Jeder Eintrag ist eine Folge der folgenden Felder: id, xoffset, yoffset, zoffset, aoffset, boffset, coffset, uoffset, voffset, woffset, Durchmesser, Vorderwinkel, Rückwinkel, Orientierung. Bei id und orientation handelt es sich um Ganzzahlen, bei den übrigen um Fließkommazahlen.

```
#!/usr/bin/env python3
# -*- coding:  utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
# das geladene Werkzeug befindet es sich in der Werkzeugtabelle an Index 0
if s.tool_table[0].id != 0: # ein Werkzeug ist geladen
    print(s.tool_table[0].zoffset)
else:
    print("Kein Werkzeug geladen")
```

toolinfo(*toolno*)

(gibt ein Diktat der Werkzeugdaten für toolno zurück)' - Eine anfängliche stat.poll() ist zur Initialisierung erforderlich. *toolno* muss größer als Null und kleiner oder gleich der höchsten verwendeten Werkzeugnummer sein. Die Wörterbucheinträge umfassen alle tooldata-Einträge: *toolno*, *pocketno*, *diameter*,*frontangle*,*backangle*,*orientation*, *xoffset*,*yoffset*, ... *woffset*, *comment* (Kommentar).
Das folgende Skript dient als Beispiel

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
toolno = 1
print(s.toolinfo(toolno))
```

```
liefert die Ausgabe:
```

```
': 0, 'xoffset': 0.0, 'yoffset': 0.0, 'zoffset': 0.18, 'aoffset': 0.0, 'boffset': 0.0,
'coffset': 0.0, 'uoffset': 0.0, 'voffset': 0.0, 'woffset': 0.0, 'comment': 'Tool_18
28Jan23:18.53.25'}
```

velocity

(returns float) - Diese Eigenschaft ist definiert, hat aber keine sinnvolle Interpretation.

Das "Achsen"-Wörterbuch

Die Achsenkonfiguration und die Statuswerte sind über eine Liste von Wörterbüchern pro Achse

verfügbar. Hier ein Beispiel, wie man auf ein Attribut einer bestimmten Achse zugreifen kann: Beachten Sie, dass viele Eigenschaften, die früher im "Achsen"-Wörterbuch standen, jetzt im "Gelenk"-Wörterbuch zu finden sind, da diese Elemente (wie z. B. das Spiel) auf nichttrivialen Kinematikmaschinen nicht zu den Eigenschaften einer Achse gehören.

max_position_limit

(returns float) - maximale Grenze (weiche Grenze) für Achsenbewegung, in Maschineneinheiten. Konfigurationsparameter, spiegelt `[JOINT___n__]MAX_LIMIT` wider.

min_position_limit

(returns float) - minimale Grenze (weiche Grenze) für Achsenbewegung, in Maschineneinheiten. Konfigurationsparameter, reflektiert `[JOINT___n__]MIN_LIMIT`.

velocity

(returns float) - aktuelle Geschwindigkeit.

Das `Gelenk`(engl. joint) Wörterbuch

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
print("Joint 1 auf Referenzposition: ", s.joint[1]["homed"])
```

Für jedes Gelenk sind die folgenden Wörterbuchschlüssel verfügbar:

backlash (engl. für Umkehrspiel)

(returns float) - Spiel in Maschineneinheiten. Konfigurationsparameter, spiegelt `[JOINT___n__]BACKLASH` wider.

enabled (engl. für aktiviert)

(gibt eine ganze Zahl zurück) - ein Wert ungleich Null bedeutet aktiviert.

fault (engl. für Fehler)

(gibt eine ganze Zahl zurück) - ein Wert ungleich Null bedeutet einen Achsverstärkerfehler.

ferror_current

(returns float) - aktueller Schleppfehler.

ferror_highmark

(returns float) - Größe des maximalen Schleppfehlers.

homed

(returns integer) - Nicht-Null bedeutet, dass die Referenzposition eingenommen wurde.

homing

(returns integer) - ungleich Null bedeutet Referenfahrt im Gange.

inpos

(gibt ganze Zahl zurück) - ungleich Null bedeutet in Position.

input

(returns float) - aktuelle Eingabeposition.

jointType

(gibt ganze Zahl zurück) - Typ des Achsenkonfigurationsparameters, entspricht `[JOINT___n__]TYPE`mit `LINEAR=1`, `ANGULAR=2`. Siehe Gelenk INI Konfiguration für Details.

max_ferror

(returns float) - maximaler Schleppfehler. Konfigurationsparameter, reflektiert `[JOINT___n__]FERROR`.

max_hard_limit

(gibt ganze Zahlen zurück) - ungleich Null bedeutet, dass der maximale harte Grenzwert überschritten wird.

max_position_limit

(gibt Float zurück) - maximaler Grenzwert (soft limit) für die Gelenkbewegung, in Maschineneinheiten. Konfigurationsparameter, spiegelt `[JOINT___n__]MAX_LIMIT`.

max_soft_limit

ungleich Null bedeutet, dass `max_position_limit` überschritten wurde, int

min_ferror

(returns float) - Konfigurationsparameter, spiegelt `[JOINT___n__]MIN_FERROR` wider.

min_hard_limit

(gibt eine ganze Zahl zurück) - ungleich Null bedeutet, dass der minimale harte Grenzwert überschritten wird.

min_position_limit

(returns float) - Mindestgrenze (weiche Grenze) für Gelenkbewegung, in Maschineneinheiten. Konfigurationsparameter, spiegelt `[JOINT___n__]MIN_LIMIT` wider.

min_soft_limit

(returns integer) - ungleich Null bedeutet, dass `min_position_limit` überschritten wurde.

output

(returns float) - befohlene Ausgabeposition.

override_limits

(gibt eine ganze Zahl zurück) - ein Wert ungleich Null bedeutet, dass die Grenzen außer Kraft gesetzt werden.

units (engl. für Einheiten)

(returns float) - Gelenkeinheiten pro mm oder pro Grad für Winkelgelenke.
(Gelenkeinheiten sind dasselbe wie Maschineneinheiten, sofern nicht anders durch den Konfigurationsparameter `[JOINT___n__]UNITS` festgelegt)

velocity

(returns float) - aktuelle Geschwindigkeit.

Das `Spindel`-Wörterbuch

brake (engl. für Bremse)

(gibt ganze Zahl zurück) - Wert des Spindelbremsflags.

direction (engl. für Richtung)

(returns integer) - Drehrichtung der Spindel mit vorwärts=1, rückwärts=-1.

enabled (engl. für aktiviert)

(gibt ganze Zahl zurück) - Wert des Flags "Spindel aktiviert".

homed

(derzeit nicht implementiert)

increasing (engl. für zunehmend)

(gibt ganze Zahl zurück) - unklar.

orient_fault

(gibt ganze Zahl zurück)

orient_state

(gibt ganze Zahl zurück)

override

(returns float) - Spindel Geschwindigkeits-Neufestsetzungs-Skala.

override_enabled

(returns boolean) - Wert der Spindel-Neufestsetzungs-flag (engl. spindle override).

speed

(liefert Gleitkommazahl) – Spindeldrehzahl in U/min. > 0: im Uhrzeigersinn, < 0: gegen den Uhrzeigersinn.
Bei aktivem G96 entspricht der Wert der maximalen Drehzahl, die durch das optionale *D*-Wort von G96 festgelegt wird, oder – falls das *D*-Wort fehlt – den Standardwerten ±1e30.

13.2.4. Vorbereitung des Sendens von Befehlen

Einige Befehle können immer gesendet werden, unabhängig von Modus und Zustand; zum Beispiel kann die Methode `linuxcnc.command.abort()` immer aufgerufen werden.

Andere Befehle können nur in einem geeigneten Zustand gesendet werden, und diese Tests können etwas knifflig sein. Zum Beispiel kann ein MDI-Befehl nur gesendet werden, wenn:

- NOTAUS (engl. ESTOP) nicht ausgelöst wurde und
- die Maschine eingeschaltet ist und
- Referenzfahrten an den Achsen durchgeführt wurden
- der Interpreter nicht läuft und
- der Modus ist auf "MDIModus" eingestellt ist

Ein geeigneter Test vor dem Senden eines MDI-Befehls durch `linuxcnc.command.mdi()` könnte sein:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
c = linuxcnc.command()

def ok_for_mdi():
    s.poll()
    return not s.estop and s.enabled and (s.homed.count(1) == s.joints) and (s
.interp_state == linuxcnc.INTERP_IDLE)

if ok_for_mdi():
    c.mode(linuxcnc.MODE_MDI)
    c.wait_complete() # warte bis mode Wechsel ausgeführt (oder der voreingestellte
timeout von 5s eingetreten)
    c.mdi("G0 X10 Y20 Z30")
```

WARNING Lese die wichtige Information zu *wait_complete()* im nachfolgenden Abschnitt zu *linuxcnc.command methods*.

13.2.5. Senden von Befehlen über *linuxcnc.command*

Initialisieren Sie vor dem Senden eines Befehls einen Befehlskanal wie folgt:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
c = linuxcnc.command()

# Anwendungsbeispiele für einige der unten aufgeführten Befehle:
c.abort()

c.auto(linuxcnc.AUTO_RUN, program_start_line)
c.auto(linuxcnc.AUTO_STEP)
c.auto(linuxcnc.AUTO_PAUSE)
c.auto(linuxcnc.AUTO_RESUME)

c.brake(linuxcnc.BRAKE_ENGAGE)
c.brake(linuxcnc.BRAKE_RELEASE)
```

```
c.flood(linuxcnc.FLOOD_ON)
c.flood(linuxcnc.FLOOD_OFF)

c.home(2)

c.jog(linuxcnc.JOG_STOP,        jjogmode, joint_num_or_axis_index)
c.jog(linuxcnc.JOG_CONTINUOUS,  jjogmode, joint_num_or_axis_index, velocity)
c.jog(linuxcnc.JOG_INCREMENT,   jjogmode, joint_num_or_axis_index, velocity, increment)

c.load_tool_table()

c.maxvel(200.0)

c.mdi("G0 X10 Y20 Z30")

c.mist(linuxcnc.MIST_ON)
c.mist(linuxcnc.MIST_OFF)

c.mode(linuxcnc.MODE_MDI)
c.mode(linuxcnc.MODE_AUTO)
c.mode(linuxcnc.MODE_MANUAL)

c.override_limits()

c.program_open("foo.ngc")
c.reset_interpreter()

c.tool_offset(toolno, z_offset,  x_offset, diameter, frontangle, backangle, orientation)
```

`linuxcnc.command` Attribute

`serial`

die Seriennummer des aktuellen Befehls

`linuxcnc.command` Methoden:

`abort()`

EMC_TASK_ABORT-Meldung senden.

`auto(int[, int])`

Ausführen, Einzelschritte ausführen, Anhalten oder Fortsetzen eines Programms.

`brake(int)`

Spindelbremse aktivieren oder lösen.

`debug(int)`

die Debug-Stufe über die Meldung EMC_SET_DEBUG einstellen.

`display_msg(string)`

sendet eine Bedienermeldung auf den Bildschirm. (max. 254 Zeichen)

`error_msg(string)`

sendet eine Bedienerfehlermeldung auf den Bildschirm. (max. 254 Zeichen)

`feedrate(float)`

den Vorschub-Override setzen, 1.0 = 100%.

`flood(int)`

Ein-/ausschalten der Kühlmittel-Flut.

Syntax

flood(command)
flood(linuxcnc.FLOOD_ON)
flood(linuxcnc.FLOOD_OFF)

Konstanten

FLOOD_ON
FLOOD_OFF

`home(int)`

ein bestimmtes Gelenk zu Referenzpunkt fahren.

`jog(command-constant, bool, int[, float[, float]])`

Syntax

jog(command, jjogmode, joint_num_or_axis_index, velocity[, distance])
jog(linuxcnc.JOG_STOP, jjogmode, joint_num_or_axis_index)
jog(linuxcnc.JOG_CONTINUOUS, jjogmode, joint_num_or_axis_index, velocity)
jog(linuxcnc.JOG_INCREMENT, jjogmode, joint_num_or_axis_index, velocity, distance)

Befehlskonstanten

linuxcnc.JOG_STOP
linuxcnc.JOG_CONTINUOUS
linuxcnc.JOG_INCREMENT

jjogmode

True

Einzelne Gelenkbewegungen anfordern (erfordert teleop_enable(0))

False

Abfrage der kartesischen Achsenkoordinaten (erfordert teleop_enable(1))

joint_num_or_axis_index

Für gemeinsames Joggen (Jogmode=1)

joint_number

Für kartesisches Joggen der Achse (jjogmode=0)

Nullbasierter Index der Achsenkoordinate in Bezug auf die bekannten Koordinatenbuchstaben XYZABCUVW (x⇒0,y⇒1,z⇒2,a⇒3,b⇒4,c⇒5,u⇒6,v⇒7,w⇒8)

load_tool_table()

die Werkzeugtabelle neu laden.

maxvel(float)

set maximum velocity

mdi(string)

einen MDI-Befehl senden. Maximal 254 Zeichen.

mist(int)

Kühlnebel ein-/ausschalten.

Syntax

mist(command)
mist(linuxcnc.MIST_ON)
mist(linuxcnc.MIST_OFF)

Konstanten

MIST_ON
MIST_OFF

mode(int)

Modus einstellen (MODE_MDI, MODE_MANUAL, MODE_AUTO).

override_limits()

Flag für die Überschreitung der Achsengrenzen setzen.

program_open(string)

eine NGC-Datei öffnen.

rapidrate()

set rapid override factor

reset_interpreter()

den RS274NGC-Interpreter zurücksetzen

set_adaptive_feed(int)

adaptiven Vorschub (engl. adaptive feed)-Flag setzen

set_analog_output(int, float)

Wert auf analogen Ausgangs-Pin legen

set_block_delete(int)

setze Block-löschen-Markierung (engl. flag)

set_digital_output(int, int)

Digitalen Ausgangspin auf Wert setzen

set_feed_hold(int)

Vorschubfreigabe ein/auseinstellen

set_feed_override(int)

Vorschub-Neufestsetzung ein/ausschalten

set_max_limit(int, float)

Einstellung der maximalen Positionsgrenze für eine bestimmte Achse

set_min_limit()

Legen Sie die minimale Positionsgrenze für eine bestimmte Achse fest

set_optional_stop(int)

optionalen Stopp ein-/ausschalten

set_spindle_override(int [, int])

Spindel-Neufestsetzung (engl. override) einstellen aktiviert. Standardmäßig ist die Spindel 0 eingestellt.

spindle(direction: int, speed: float=0, spindle: int=0, wait_for_speed: int=0)

- Direction (engl. für Richtung): [SPINDLE_FORWARD, SPINDLE_REVERSE, SPINDLE_OFF, SPINDLE_INCREASE, SPINDLE_DECREASE, oder SPINDLE_CONSTANT]
- Speed: Drehzahl in U/min (engl. RPM), Standardwert ist 0.
- Spindle: Spindelnummer für den Befehl ist standardmäßig 0.
- Wait_for_speed: wenn auf 1 gesetzt warten Bewegung auf die Geschwindigkeit bevor sie fortgesetzt werden, die Standardeinstellung verlang dies nicht.

WARNING MDI-Befehle ignorieren dies. "S1000" danach schaltet die Spindel aus.

text_msg(string)

sendet eine Betreiber-Textnachricht auf den Bildschirm (max. 254 Zeichen).

```
#!/usr/bin/env python3
import linuxcnc
c = linuxcnc.command()

# Erhöhen Sie die Drehzahl der Spindel 0 um 100 Umdrehungen pro Minute. Die Spindel muss
zuerst eingeschaltet werden.
c.spindle(linuxcnc.INCREASE)

# Erhöhen Sie die Drehzahl von Spindel 2 um 100 Umdrehungen pro Minute. Die Spindel muss
zuerst eingeschaltet werden.
c.spindle(linuxcnc.SPINDLE_INCREASE, 2)

# Set speed of spindle 0 to 1024 U/min.
c.spindle.(linuxcnc.SPINDLE_FORWARD, 1024)

# Drehzahl der Spindel 1 auf -666 U/min setzen.
```

```
c.spindle.(linuxcnc.SPINDLE_REVERSE, 666, 1)

# Spindel 0 stoppen.
c.spindle.(linuxcnc.SPINDLE_OFF)

# Spindel 0 explizit stoppen.
c.spindle.(linuxcnc.SPINDLE_OFF, 0)
```

spindleoverride(float [, int])

Spindel-Neufestsetzung (engl. override)-Faktor einstellen. Bezieht sich wenn nicht anders angegeben auf Spindel 0.

state(int)

Setzt den Maschinenzustand. Der Maschinenzustand sollte STATE_ESTOP, STATE_ESTOP_RESET, STATE_ON, oder STATE_OFF sein.

task_plan_sync()

Nach Beendigung dieses Aufrufs wird die VAR-Datei auf der Festplatte mit den aktuellen Werten des Interpreters aktualisiert.

teleop_enable(int)

Teleop-Modus aktivieren/deaktivieren (für gemeinsames Joggen deaktivieren).

tool_offset(int, float, float, float, float, float, int)

Den Werkzeugversatz einstellen. Siehe Anwendungsbeispiel oben.

traj_mode(int)

Trajektorienmodus einstellen. Der Modus ist einer von MODE_FREE, MODE_COORD oder MODE_TELEOP.

unhome(int)

Referenzpunkt eines bestimmten Gelenks auflösen (engl. unhome).

wait_complete([float])

Auf die Beendigung des letzten Befehls warten. Akzeptiert ein optionales Argument zur Festlegung des timeouts (Zeitüberschreitung).
Dessen Standardwert ist 5 Sekunden wenn nicht angegeben.
Gibt -1 bei Zeitüberschreitung zurück.
Gibt RCS_DONE oder RCS_ERROR zurück entsprechend dem Status der Befehlsausführung.
Bemerke dass die Ausführung von Python blockiert bis diese Funktion ausgeführt wurde.

13.2.6. Lesen des Fehlerkanals

Um Fehlermeldungen zu behandeln, stellen Sie eine Verbindung zum Fehlerkanal her und pollen (durch Aufruf von poll()) Sie ihn regelmäßig.

Beachten Sie, dass der NML-Kanal für Fehlermeldungen über eine Warteschlange verfügt (anders als die Befehls- und Statuskanäle), was bedeutet, dass der erste Empfänger einer Fehlermeldung diese aus der Warteschlange löscht; ob ein anderer Empfänger einer Fehlermeldung (z. B. AXIS) die Meldung sieht,

hängt vom Timing ab. Es wird empfohlen, nur eine Fehlerkanal-Leser-Task in einem Setup zu haben.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
e = linuxcnc.error_channel()

error = e.poll()

if error:
    kind, text = error
    if kind in (linuxcnc.NML_ERROR, linuxcnc.OPERATOR_ERROR):
        typus = "error"
    else:
        typus = "info"
    print(typus, text)
```

13.2.7. Reading INI file values using `linuxcnc.ini`

The `linuxcnc.ini` module includes all functions to read values and information from INI-files.

Reading an INI-file is done by creating an instance of the module class:

```
import linuxcnc

try:
    ini = linuxcnc.ini("/path/to/ini-file.ini")
except linuxcnc.error:
    print("Ini-file not found or invalid")
```

The `ini` instance can then be used to extract values and other information from the INI-file.

IMPORTANT

You should not roll your own type conversion functions to read boolean, integer or floating point values. The INI-file parser and Python wrapper already include these functions.

Using the built-in functions guarantees consistent interpretation of values, also across locales. There are also helpers for enumerated types to read and convert them correctly and consistently.

`linuxcnc.ini` Methods

All methods will throw an error on the following conditions:

- `TypeError`, `ValueError`, `UnicodeError`, `OverflowError`: an argument was of the wrong type or value,
- `linuxcnc.error`: the *num* argument, if given, was less or equal zero (0).

NOTE

All methods that retrieve a `[SECTION]VARIABLE` value can provide an empty string ("")

> for the *section*. In that case, the first matching *variable* found in any section of the INI-file is used or returned.
>
> This behavior is for compatibility only. You should not rely on this property.

For all methods, the *fallback*= value, if given, may be of any type. It is returned verbatim if the method otherwise would have failed or returned `None`.

Conversion to both integer and floating point will ignore trailing content if it is separated by whitespace.

Beispiel

```
[BLUEBERRY]
PIE = 3.14159265 This is _/*very*/_ bad style but accepted
FAIL= 0.12345#Warning will be issued
```

The above example will correctly convert when using `ini.getreal("BLUEBERRY","PIE")` and the trailing content is ignored. A warning message (`inifile:line: warning: Trailing character(s)...`) will be reported on `ini.getreal("BLUEBERRY","FAIL")`.

Method details

ini = linuxcnc.ini(*filename*:string)

Creates an INI-file instance reading INI-file *filename*. Throws a `linuxcnc.error` if the file does not exist or cannot be parsed.

bool = ini.hassection(*section*:string)

Returns a boolean indicating whether or not the given *section* was found in the ini-file.

bool = ini.hasvariable(*section*:string, *variable*:string [, *num*:int])

Returns a boolean indicating whether or not the given *variable* in *section* was found in the ini-file. The first occurrence of the *variable* name in any section will be searched if the *section* name is empty. The optional *num* argument may be used to test whether the num'th *variable* exists in the *section*.

bool|None = ini.getbool(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Returns the value of the *variable* converted to boolean if it was a valid boolean. The return value is `None` if the *variable* was not found or an invalid value was detected and no *fallback*= was provided. The optional *num* argument may be used to select the num'th *variable* of that name in the *section*.

int|None = ini.getsint(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Returns the value of the *variable* converted to signed integer if it was a valid integer. The return value is `None` if the *variable* was not found or an invalid value was detected and no *fallback*= was provided. The optional *num* argument may be used to select the num'th *variable* of that name in the *section*.

int|None = ini.getint(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Alias of `ini.getsint()`.

int|None = ini.getuint(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Returns the value of the *variable* converted to unsigned integer if it was a valid unsigned integer. The return value is `None` if the *variable* was not found or an invalid value was detected and no *fallback*= was provided. The optional *num* argument may be used to select the num'th *variable* of that name in the *section*.

float|None = ini.getreal(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Returns the value of the *variable* converted to floating point real if it was a valid real. The return value is `None` if the *variable* was not found or an invalid value was detected and no *fallback*= was provided. The optional num argument may be used to select the num'th *variable* of that name in the *section*.

float|None = ini.getfloat(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Alias of `ini.getreal()`.

string|None = ini.getstring(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Returns the value of the *variable* as a string if it exists. The return value is `None` if the *variable* was not found and no *fallback*= was provided. The optional *num* argument may be used to select the num'th *variable* of that name in the *section*.

list = ini.getsections()

Returns a list of *section* names.

list(name,value) = ini.getvariables([*section*:string])

Returns a list of (name,value) tuples of all variables in the named *section*, or the variables from all sections if the *section* name is not specified.

string|None = ini.find(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Alias of `ini.getstring()`.

list = ini.findall(*section*:string [,*variable*:string])

Find value(s) from named *section* as a list matching the optional *variable* name.

tuple(filename,lineno) = ini.lineof(*section*:string, *variable*:string [, *num*:int])

Returns a tuple with the filename and line number of *num*'th *variable* in the *section*. The first matching *section variable* is returned if *num* is not provided. The tuple (None, None) is returned if the *variable* is not found.

float|None = ini.maplinearunits(*enumstr*:string [, *fallback*=])

Take the *enumstr* enumeration string argument and try to convert. Returns the value associated with enumerated type defined by [*mm*, *metric*, *in*, *inch*, *imperial*]. Returns `None` if the *enumstr* contains an invalid string and *fallback*= is not specified.

float|None = ini.mapangularunits(*enumstr*:string [, *fallback*=])

Take the *enumstr* enumeration string argument and try to convert. Returns the value associated with enumerated type defined by [*deg*, *degree*, *grad*, *gon*, *rad*, *radian*]. Returns `None` if the *enumstr* contains an invalid string and *fallback*= is not specified.

int|None = ini.mapjointtype(*enumstr*:string [, *fallback*=])

Take the *enumstr* enumeration string argument and try to convert. Returns the value associated with enumerated type defined by [*LINEAR*, *ANGULAR*]. Returns `None` if the *enumstr* contains an invalid string and *fallback*= is not specified.

float|None = ini.getlinearunits(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Get the ini *variable* from the *section* and convert the enumerated type. The optional *num* argument may be used to select the num'th *variable* of that name in the *section*. Returns the value associated with the enumerated type defined by [*mm*, *metric*, *in*, *inch*, *imperial*]. Returns `None` if the *variable* is not found and *fallback*= is not specified.

float|None = ini.getangularunits(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Get the ini *variable* from the *section* and convert the enumerated type. The optional *num* argument may be used to select the num'th *variable* of that name in the *section*. Returns the value associated with the enumerated type defined by [*deg*, *degree*, *grad*, *gon*, *rad*, *radian*]. Returns `None` if the *variable* is not found and *fallback*= is not specified.

int|None = ini.getjointtype(*section*:string, *variable*:string [, *num*:int] [, *fallback*=])

Get the ini *variable* from the *section* and convert the enumerated type. The optional num argument may be used to select the num'th *variable* of that name in the *section*. Returns the value associated with enumerated type defined by [*LINEAR*, *ANGULAR*]. Returns `None` if the *variable* is not found and *fallback*= is not specified.

`linuxcnc.ini` Examples

Reading values from an INI file through the `linuxcnc.ini` object:

```
#!/usr/bin/env python3
# run as:
# python3 ini-example.py ~/emc2-dev/configs/sim/axis/axis_mm.ini

import sys
import linuxcnc

fname = sys.argv[1] if len(sys.argv) > 1 else "<missing filename>"
try:
    inifile = linuxcnc.ini(fname)
except linuxcnc.error as detail:
    print("{}: error: {}".format(fname, detail))
    sys.exit(1)

# inifile.getstring() returns None if the key wasn't found - the
# following idiom is useful for setting a default value:
machine_name = inifile.getstring("EMC", "MACHINE", fallback="unknown")
print("machine name:", machine_name)

# inifile.findall() returns a list of matches, or an empty list
# if the key wasn't found:
extensions = inifile.findall("FILTER", "PROGRAM_EXTENSION")
print("extensions  :", extensions)
```

```
# Other examples:
realval = inifile.getreal("AXIS_X", "MAX_VELOCITY", fallback=5.0)
boolval = inifile.getbool("JOINT_0", "HOME_USE_INDEX", fallback=False)

# None is returned without fallback= if the variable was not found
intval  = inifile.getint( "KINS", "JOINTS")
if intval is None:
    print("Error: [KINS]JOINTS not defined or an invalid integer")
else:
    print("[KINS]JOINTS:", intval)
```

Oder für die selbe INI-Datei wie LinuxCNC:

```
#!/usr/bin/env python3
# run as:
# python3 ini-example2.py

import linuxcnc

stat = linuxcnc.stat()
stat.poll()

inifile = linuxcnc.ini(stat.ini_filename)

# Siehe obiges Beispiel für die Verwendung des 'inifile' Objekts
```

13.2.8. Der Typ `linuxcnc.positionlogger`

Einige Verwendungshinweise können von *src/emc/usr_intf/gremlin/gremlin.py* entnommen werden.

Members

`npts`

Anzahl der Punkte.

Methoden

`start(float)`

den Positionslogger starten und alle ARG-Sekunden ausführen

`clear()`

den Positionslogger löschen

`stop()`

den Positionslogger anhalten

`call()`

Plotte jetzt den Backplot.

`points()`

Return `(bytes, npts, is_xyuv)`: a copy (taken under the sampler lock) of the logged point buffer, the number of points, and whether the machine is a foam (XY/UV) cutter. Each point is a C `logger_point` — 3 float32 (x, y, z), 4 uint8 RGBA, 3 float32 (rx, ry, rz or u, v, w), 4 uint8 RGBA — so the buffer is `npts * 32` bytes. Added for the OpenGL 3.3 core renderer to upload the backplot to a GPU buffer; see `rs274.glcanon` for the numpy layout.

`last([int])`

Gibt den letzten Punkt auf dem Plot oder keinen (als Python Ausdruck `None`) zurück

13.3. Das HAL Python-Modul

This documentation describes the `hal` Python module, which provides a Python API for creating and accessing HAL pins and signals.

IMPORTANT

The classes inside the Python module `hal` are layered on top of the module implementation called `_hal`. Many features described in this document are ***only*** available in the `hal` Python module.

You should only import and access the `hal` module API as described in this document. The base-class methods, members, properties and values may change without notice.

13.3.1. Grundlegende Verwendung (engl. basic usage)

Simple example creating component and pins

```
#!/usr/bin/env python3
import hal
import time

comp = hal.component("multiply")
comp.newpin("in-a", hal.Type.REAL, hal.Dir.IN)
comp.newpin("in-b", hal.Type.REAL, hal.Dir.IN)
comp.newpin("out", hal.Type.REAL, hal.Dir.OUT)
comp.ready()

try:
    while True:
        comp['out'] = comp['in-a'] * comp['in-b']
        time.sleep(0.001)
except KeyboardInterrupt:
    raise SystemExit
```

13.3.2. Class `hal`

hal enumerations and constants

There are many constants used to indicate type, direction, levels and much more. These are usually integer values with a symbolic name. It is preferable in code to use the symbolic name for readability and portability.

Enumerations have been added to aid the readability even more. They use the Python IntEnum base class. The IntEnum values behave as numerical values, but add clear visibility features for readability. The old defined constants are still supported for backwards compatibility.

Beispiel:

```
comp.newpin("in-a", hal.Type.REAL, hal.Dir.IN)
```

HAL types

The `hal.Type` enum is used to specify the type of pin, parameter and signal:

- `hal.Type.BOOL` - A boolean using `True` and `False`
- `hal.Type.SINT` - A signed quantity with range -2^{63}...$+2^{63}-1$
- `hal.Type.UINT` - An unsigned quantity with range 0...$+2^{64}-1$
- `hal.Type.REAL` - A floating point quantity of range $\pm 1.80 \times 10^{308}$
- `hal.Type.PORT` - An opaque quantity representing a communication channel (pins only)

There are aliases in the `hal.Type` enum for all types with the `HAL_` prefix (e.g. `hal.Type.HAL_BOOL`, etc.).

NOTE

The constants previously used (`hal.HAL_BIT`, `hal.HAL_FLOAT`, etc.) are deprecated and emit a warning. You should upgrade your code to use the enumerated types with the new names instead.

```
# Old (deprecated) style:
comp.newpin("in-a", hal.HAL_FLOAT, hal.HAL_IN)

# New style:
comp.newpin("in-a", hal.Type.REAL, hal.Dir.IN)
```

HAL direction

Just like types, `hal.Dir` enum is the enumeration to specify direction of pins and parameters. The direction for both pin and parameter direction is unified in one enumerated type:

- `hal.Dir.IN` - An input pin
- `hal.Dir.OUT` - An output pin
- `hal.Dir.IO` - A bidirections pin
- `hal.Dir.RW` - A read/write parameter

- `hal.Dir.RO` - A read-only parameter

Also the direction enums are available with the `HAL_` prefix (like `hal.Dir.HAL_IN`).

Old style HAL constants (deprecated)

Old style constants (deprecated, please use `hal.Type.XXXX`): `hal.HAL_BOOL`, `hal.HAL_SINT`, `hal.HAL_UINT`, `hal.HAL_REAL`, `hal.HAL_PORT`.

Deprecated constants (replaced with new names, please use `hal.Type.XXXX`): `hal.HAL_BIT`, `hal.HAL_S64`, `hal.HAL_U64`, `hal.HAL_FLOAT`.

Old style deprecated pin direction constants (deprecated, please use `hal.Dir.YYY`): `hal.HAL_IN`, `hal.HAL_OUT`, `hal.HAL_IO`.

Old style deprecated parameter access constants (deprecated, please use `hal.Dir.YYY`): `hal.HAL_RO`, `hal.HAL_RW`.

Message level constants

- `hal.MSG_NONE` - No messages at all
- `hal.MSG_ERR` - Only errors
- `hal.MSG_WARN` - Both warnings and errors
- `hal.MSG_INFO` - Both informational, warning and error messages
- `hal.MSG_DBG` - Additionally include debugging information
- `hal.MSG_ALL` - Print all messages encountered, disregarding level

Realtime type enumerations

The enumerated type `hal.RTType` is used in the `hal.get_realtime_type()` method.

- `hal.RTType.UNINITIALIZED` - Real time module not running
- `hal.RTType.NONE` - No realtime available
- `hal.RTType.UNKNOWN` - Only used when LINUXCNC_FORCE_REALTIME=1 is set. Unknown, no PREEMPT_DYNAMIC but SCHED_FIFO is available. Not recommended.
- `hal.RTType.PREEMPT_DYNAMIC` - Preempt dynamic: available in vanilla kernel. Only used when LINUXCNC_FORCE_REALTIME=1 is set. Not recommended.
- `hal.RTType.PREEMPT_RT` - Preempt RT
- `hal.RTType.RTAI` - RTAI kernel mode
- `hal.RTType.LXRT` - LXRT, userspace implementation for RTAI
- `hal.RTType.XENOMAI` - Xenomai 3
- `hal.RTType.XENOMAI_EVL` - Xenomai 4 aka Xenomai EVL

The realtime enumerations are also available as constants under the `hal.REALTIME_TYPE_xxx` name

where `xxx` is one of the above.

Component type enumeration

The enumerated type `hal.CompType` is used in `hal.query.comp()` and `hal.query.comps()`.

- `hal.CompType.REALTIME` - a realtime component
- `hal.CompType.USER` - a user non-realtime component
- `hal.CompType.OTHER` - (HAL internal value; should normally not be visible)

Other constants

System information:

- `hal.is_kernelspace` - One (1) if RTAPI runs in the kernel, otherwise zero (0)
- `hal.is_userspace` - Inverted `hal.is_kernelspace`
- `hal.kernel_version` - A string specifying the real-time kernel version if `hal.is_kernelspace` is one. Otherwise it specifies `"Not Available"`.
- `hal.is_rt` - One (1) if the system runs in real-time, otherwise zero (0) **DEPRECATED**: Use `lcnc_realtime.verify()`
- `hal.is_sim` - Inverted `hal.is_rt` **DEPRECATED**: Use `lcnc_realtime.verify()`

hal methods

hal.is_initialized() → bool

Returns a boolean to indicate whether hal is initialized. This methods should **always** return `True` because the HAL module is automatically initialized at import time.

Beispiel:

```
import hal

assert hal.is_initialized(), "fatal: HAL failed to initialize at import"
```

hal.get_realtime_type() → hal.RTType

Returns the type of the running realtime system. Might return `hal.RTType.UNINITIALIZED` if `rtapi_app` is not running. See realtime type constants. See also LinuxCNC Realtime check.

hal.component_exists(*name*:string) → bool

Returns a boolean to indicate whether or not the specified component exist at this time.

hal.component_is_ready(*name*:string) → bool

Returns a boolean to indicate whether or not the specified component is in the ready state. Also returns False if the component does not exist.

Beispiel:

```
if not hal.component_is_ready("testpanel"):
    compmsg = "ready" if hal.component_exists("testpanel") else "loaded"
    print("Expected component 'testpanel' to be {}".format(compmsg))
    os.exit(1)
```

hal.set_msg_level(*lvl*:int)

Set the message level that controls the amount of information being printed and forwarded from real-time components. The *lvl* argument must be one of the message constants.

hal.get_msg_level() → int

Return the current message level. See *hal.set_msg_level()* and message constants for list of possible returned values.

hal.new_sig(*name*:string, *type*:enum) → bool

Create a new signal (net) called *name*. The signal can carry information of *type* content. Returns `True` on success.

Beispiel:

```
if not hal.new_sig("signalname", hal.Type.BOOL):
    ...handle error...
```

hal.connect(*pinname*:string, *signame*:string) → bool

Connect the pin *pinname* to signal *signame*. Both signal and pin must exist and both pin and signal must be of the same type. Returns `True` on success.

Beispiel:

```
if not hal.connect("mycomp.pinname", "signalname"):
    ...handle error...
```

hal.disconnect(*pinname*:string, *signame*:string) → bool

Disconnect the pin *pinname* from signal *signame*. Both signal and pin must exist. Returns `True` on success.

Beispiel:

```
if not hal.disconnect("mycomp.pinname"):
    ...handle error...
```

hal.pin_has_writer(*pinname*:string) → bool

Returns `True` if pin with name *pinname* is attached to a signal and there is at least one writer. Otherwise, `False` is returned.

Beispiel:

```
if hal.pin_has_writer("mycomp.0.pin.02"):
    print("Pin has writer")
else:
```

```
    print("Pin has no writer or no signal attached")
```

hal.set_p(*name*:string, *value*:mixed)

Sets the pin or param called *name* to *value*. The *name* is the full name of the pin or param. The search order is pin names first, then parameter names.
Throws a `RuntimeError` exception if the *name* is not found.
The type of *value* depends on the type of the pin or param. Integer scalar types may use integers or a textual representation of an integer to set the value. Floating point type may use both integer, floating point and textual representation thereof to set the value. Booleans accept `True`, `False` and map the integer value zero (0) to false. The exact floating point value of zero (0.0) also maps for false. Booleans may also be of text "0", "1", "on", "off", "true", "false", "yes" or "no". Textual representations are case insensitive.

Beispiel:

```
hal.set_p("mycomp.0.bool", True)
hal.set_p("mycomp.0.real", 99.99)
```

hal.set_s(*name*:string, *value*:mixed)

Sets the signal (net) called *name* to *value*. The same rules for *value* apply to *set_s()* as to *set_p()*.

The *set_s()* method has one special case when the signal is of type `hal.Type.PORT` and it is fully connected. In that case, the call uses the *value* to set the port's queue size and it must be a positive integer. See below on configuring a port.

hal.get_p(*name*:string) → bool|int|float|None

Returns the value of the pin or param with *name*. If *name* refers to a pin and that pin is connected, then the connected signal's value is returned. Boolean types return `True` or `False`. Integer scalar types return an integer. Floating point types return a float. The value `None` is returned if no pin or param is found by that name.

hal.get_s(*name*:string) → bool|int|float|None

Returns the value of the signal with *name*. Boolean types return `True` or `False`. Integer scalar types return an integer. Floating point types return a float. The value `None` is returned if no signal is found by that name.

hal.get_value(*name*:string) → bool|int|float

Returns the value of the pin, param or signal with *name*, searched in that order. Boolean types return `True` or `False`. Integer scalar types return an integer. Floating point types return a float. A `RuntimeError` exception is thrown if no pin, param or signal is found by that name.

Beispiel:

```
value = hal.get_value("iocontrol.0.emc-enable-in")
```

hal.get_info_pins() → list(dict)

DEPRECATED: replaced by `hal.query.pins()`.
Returns a list of dictionary tuples as in `{"NAME":"pinname", "VALUE":<bool|int|float>,`

`"TYPE":<int>, "DIRECTION":<int>}`.

Beispiel:

```
# Old style (deprecated):
for i in hal.get_info_pins():
    print(i['NAME'], i['TYPE'], i['DIRECTION'], i['VALUE'])

# New style (see hal.query sub-module below):
for pinname,pindetail in hal.query.pins().items():
    print(pinname, pindetail['type'], pindetail['dir'], pindetail['value'])
```

hal.get_info_params() → list(dict)

DEPRECATED: replaced by `hal.query.params()`.
Returns a list of dictionary tuples as in `{"NAME":"paramname", "VALUE":<bool|int|float>, "TYPE":<int>, "DIRECTION":<int>}`.

hal.get_info_signals() → list(dict)

DEPRECATED: replaced by `hal.query.signals()`.
Returns a list of dictionary tuples as in `{"NAME":"signame", "VALUE":<bool|int|float>, "TYPE":<int>, "DRIVER":"name"|None}`.

Beispiel:

```
# New style (see hal.query sub-module below)
for signame,sigdetail in hal.query.signals().items():
    print(signame, sigdetail['type'], sigdetail['value'], sigdetail['driver'])
```

13.3.3. Class `hal.component`

component methods

comp = hal.component(*name*:string [, *prefix*:string])

The component itself is created by a call to the constructor `hal.component`. The arguments are the HAL component *name* and (optionally) the *prefix* used for pin and param names. If the prefix is not specified, the component name is used.

Beispiel:

```
comp = hal.component("passthrough")
```

pin = comp.newpin(*name*:string, *type*:enum, *io*:enum)

Create new pin with actual name `prefix.__name__`. The pin *type* must be one of the types described above. The *io* specifies the direction of the pin. Throws a `ValueError` exception if *name* already exists.

Beispiel:

```
p_in = comp.newpin("in", hal.Type.FLOAT, hal.Dir.IN)
```

param = comp.newparam(*name*:string, *type*:enum, *access*:enum)

Create new parameter with actual name `prefix.__name__`. The pin *type* argument must be one of the types described above. Parameters cannot be of type `hal.Type.PORT`. The *access* argument specifies the allowed param access. Throws a `ValueError` exception if *name* already exists.

Beispiel:

```
p_bloop = comp.newparam("bloop", hal.Type.FLOAT, hal.Dir.RO)
```

comp.getitem(*name*:string) → object

Return the pin or param item object *name* previously created with *comp.newpin()* or *comp.newparam()*. Throws an `AttributeError` exception if no pin or param named *name* is found. Use *comp.getpin()* or *comp.getparam()* to find the specific type.

comp.getpin(*pinname*:string) → object

Return the pin item object *pinname* previously created with *comp.newpin()*. Throws an `AttributeError` exception if no pin names *pinname* is found. A param called *pinname* will not be found and throws an exception. Use *comp.getitem()* to find either.

comp.getparam(*paramname*:string) → object

Return the param item object *paramname* previously created with *copm.newparam()*. Throws an `AttributeError` exception if no pin names *paramname* is found. A pin called *paramname* will not be found and throws an exception. Use *comp.getitem()* to find either.

comp.getpins() → dict

Returns a dictionary all pin and param names and their values. The pin or param name is the dictionary key.

comp.ready()

Sagt dem HAL-System, dass die Komponente initialisiert ist. Sperrt das Hinzufügen von Pins.

comp.unready()

Allows a component to add pins after *ready()* has been called. One should call *ready()* on the component when done.

comp.getprefix() → string

Returns the current component's prefix used when creating pins and params. It defaults to the component name when not set in the constructor or by *setprefix*.

comp.setprefix(*prefix*:string)

Set the prefix used when creating pins and params. The *prefix* is used to create pins and params called `__prefix__.name` and defaults to the components name.

13.3.4. Class `hal.stream`

The streamer class enables sending typed value blocks, samples of data, between real-time and non-real-time. The values carried in a stream have the same types as pins and params. A stream can source data

from pins or sink data into pins using ready made components. The stream carries values through a FIFO queue up to a specified depth. The stream allocates and uses a shared memory segment that is not part of HAL memory and thus does not burden HAL memory usage. Each data sample may contain up to twenty (20) values. The type of the values in a sample must be configured and specified in a type string. The type string consists of the following characters (case insensitive):

- `b` - Boolean (bool)
- `s`, `l` - Signed 64-bit (sint)
- `u`, `k` - Unsigned 64-bit (uint)
- `f`, `r` - Floating point (real)

Two components are available, *streamer* and *sampler*. The *streamer* component writes a set of pins from non-real-time data pushed into a stream. The *sampler* component samples a set of pins in real-time and makes them available in a stream.

NOTE

If you use this Python `hal` module interface for both read and write on the same stream, then you must create the stream with *depth* and *type string* before you can attach to it. Instantiating with depth and type string *creates* the stream. Instantiating without depth *attaches* to the stream.

Both the *sampler* and *streamer* components create the stream. You only need to attach to it from the Python `hal` module.

IMPORTANT

A stream can only have one (1) reader and one (1) writer. The stream queue is not designed to support operation with multiple readers or writers.

Example sampler - Python sample reader:

```
import hal
import time

comp = hal.component("samplereader")
# Attach to the sampler stream
samplereader = hal.stream(comp, hal.sampler_base, "bffs")
# ...
comp.ready()
# ...

hal.set_value("sampler.0.enable", True) # Start streaming samples

while True:
    while samplereader.readable:
        print(samplereader.read())
    time.sleep(0.001) # Don't busy-loop
```

Example sampler - hal-file sampler:

```
loadrt sampler depth=100 cfg=bffs

# Disable sampler before adding function to the thread or it would start
```

```
# sampling immediately and could fill the queue before we start reading.
setp sampler.0.enable 0

addf sampler.0 servo-thread

net sample-jogger  motion.jog-is-active sampler.0.pin.0
net sample-joint-0 joint.0.pos-fb       sampler.0.pin.1
net sample-joint-1 joint.1.pos-fb       sampler.0.pin.2
net sample-line    motion.program-line  sampler.0.pin.3
```

stream constants

hal.sampler_base

The *sampler* component's (sampler.c) shared memory ID for stream communication.

hal.stream_base

The *streamer* component's (streamer.c) shared memory ID for stream communication.

stream methods

stream = hal.stream(*comp*:object, *key*:int, *depth*:int, *typestr*:string)

Create a stream for the component *comp* using the shared memory identifier *key*. The stream's queue size is allocated for *depth* samples of *typestr* format. See the type string list for type meaning.

stream = hal.stream(*comp*:object, *key*:int [, *typestr*:string])

Attach to a stream for the component *comp* where the shared memory location is identified by *key*. If the optional *typestr* is provided, then it will be checked against the existing stream's configuration. See the type string list for type meaning.

stream.read() → tuple|None

Returns a tuple of sample data from the queue. `None` is returned if no samples were available.

stream.write(*sample*:tuple)

Writes the *sample* argument to the stream queue. The *sample* must contain the correct number of elements and match the types (or be convertible) of the stream's configuration. An `IOError` exception is thrown if the sample could not be written to the queue.

stream.readable() → bool

Returns `True` if there are samples available for reading in the stream queue or `False` if not.

stream.writable() → bool

Returns `True` if there is space available in the stream queue to hold more samples or `False` if not.

stream.depth() → int

Returns the number of currently available samples for read in the queue.

stream.maxdepth() → int

Returns the queue size as set when the stream was created (and cannot be changed).

stream.element_types() → int

Return a bytes object with the format type string that was used to create the stream. See type string list for individual elements and type meaning.

stream.num_underruns() → int

Returns the number of times *stream.read()* was called when no samples were available from the queue.

stream.num_overruns() → int

Returns the number of times *stream.write()* was called when no samples could be stored in the queue.

stream members

stream.sampleno

The last successfully read sample ID number as counted by the stream functions.

13.3.5. Sub-module `hal.query`

The `hal.query` sub-module is for getting information about all of HAL's internal structures. These include:

- Pins
- Parameter
- Signale
- Komponenten
- Funktionen
- Threads

Each category can be queried by name or ID to get information about the specific item or you can get them all. The methods in this sub-module return a dictionary with all available information.

hal.query methods

hal.query.pin(*name*:string) → dict|None

Retrieve all information of the pin *name*. Returns `None` if the *name* was not found. Otherwise, returns a dictionary with the pin information:

```
result = hal.query.pin("my.pin.name")
result = {
    "haltype" : "pin",
    "name"    : "my.pin.name",
    "type"    : hal.Type.<BOOL,REAL,SINT,UINT,PORT>,
    "dir"     : hal.Dir.<IN,OUT,IO>,
    "value"   : <bool, int or float>,
    "alias"   : "alias.name"|None,
    "signal"  : "signal.name"|None,
```

```
    "comp"    : "component-name",
    "comp_id" : <int>
}
```

hal.query.param(*name*:string) → dict|None

Retrieve all information of the parameter *name*. Returns `None` if the *name* was not found. Otherwise, returns a dictionary with the parameter information:

```
result = hal.query.param("my.param.name")
result = {
    "haltype" : "parameter",
    "name"    : "my.param.name",
    "type"    : hal.Type.<BOOL,REAL,SINT,UINT>,
    "dir"     : hal.Dir.<RO,WR>,
    "value"   : <bool, int or float>,
    "alias"   : "alias.name"|None,
    "comp"    : "component-name",
    "comp_id" : <int>
}
```

hal.query.signal(*name*:string) → dict|None

Retrieve all information of the signal *name*. Returns `None` if the *name* was not found. Otherwise, returns a dictionary with the signal information:

```
result = hal.query.signal("my.signal.name")
result = {
    "haltype" : "signal",
    "name"    : "my.signal.name",
    "type"    : hal.Type.<BOOL,REAL,SINT,UINT,PORT>,
    "value"   : <bool, int or float>,
    "writers" : <int>,
    "readers" : <int>,
    "bidirs"  : <int>,
    "driver"  : "driver.pin.name"|None
}
```

The `readers`, `writers` and `bidirs` indicate how many pins are connected. There can only be one `writers` and many `readers`. Or, there can be many `bidirs` and many `readers`. The `driver` is the connected pin name of type `hal.Dir.OUT` or `None` if `writers` is zero.

hal.query.comp(*name*:string) → dict|None

dict|None = hal.query.comp(*id*:int)

Retrieve all information of the component *name* or by integer component *id*. Returns `None` if the *name* or *id* was not found. Otherwise, returns a dictionary with the component information:

```
result = hal.query.comp("component-name")
result = {
    "haltype" : "component",
    "name"    : "component-name",
    "type"    : hal.CompType.<REALTIME,USER,OTHER>
    "id"      : <int>
```

```
    "pid"     : <int>|0
    "ready"   : True|False,
    "insmod"  : "comp command line options"
}
```

The `pid` field is set to zero (0) for realtime components and the process ID for user-space components. The `insmod` field is the loadrt command line (excluding loadrt) and is only set for realtime components. The `type` field is currently only either `hal.CompType.REALTIME` or `hal.CompType.USER`.

hal.query.funct(*name*:string) → dict|None

Retrieve all information of the function *name*. Returns `None` if the *name* was not found. Otherwise, returns a dictionary with the function information:

```
result = hal.query.funct("my.funct.name")
result = {
    "haltype"   : "function",
    "name"      : "my.funct.name",
    "comp"      : "component-name",
    "comp_id"   : <int>,
    "users"     : <int>,
    "reentrant" : True|False
}
```

The `users` field indicates how many threads use this function. Only functions that have `reentrant` set to `True` can have more than one user.

hal.query.thread(*name*:string) → dict|None

Retrieve all information of the thread *name*. Returns `None` if the *name* was not found. Otherwise, returns a dictionary with the thread information:

```
result = hal.query.thread("my.thread.name")
result = {
    "haltype"   : "thread",
    "name"      : "my.thread.name",
    "comp"      : "component-name",
    "comp_id"   : <int>,
    "priority"  : <int>,
    "period"    : <int>, # in nanoseconds
    "functions" : (
        {
            "haltype" : "threadfunction",
            "name"    : "my.funct.name",
            "index"   : <int>,
            "is_init" : True|False
        },
        ...
    )
}
```

The `functions` field is a list of dictionaries. Each entry describes a function that runs in the thread.

The order of execution within the thread is indicated by zero-based `index`. Any function in the `functions` list should have the corresponding `users` field on a `hal.query.funct(name)` set.
The `is_init` field can only contain `True` if the thread was setup but never run. Init functions are automatically removed once they have executed once and never show up again.

hal.query.signalpins(*name*:string) → dict

Retrieve information about all pins connected to signal *name*. Returns `None` if the signal *name* is not found. The dictionary returned may be empty and has the format:

```
result = hal.query.signalpins("my.signal")
result = {
    "pin-r" : { "haltype": "pin", "name": "pin-r", "dir": hal.Dir.IN  "signal":
"my.signal", ...},
    "pin-w" : { "haltype": "pin", "name": "pin-w", "dir": hal.Dir.OUT "signal":
"my.signal", ...},
    "pin-x" : { "haltype": "pin", ...},
    ...
}
```

See `hal.query.pin()` for description for pin dictionary details.

hal.query.pins() → dict

Retrieve information about all pins. The dictionary returned has the format:

```
{
    "pin-a" : { "haltype": "pin", "name": "pin-a", "type":...},
    "pin-b" : { "haltype": "pin", ...},
    ...
}
```

See `hal.query.pin()` for description of the pin dictionary details.

hal.query.params() → dict

Retrieve information about all params. The dictionary returned has the format:

```
{
    "param-a" : { "haltype": "parameter", "name": "param-a", "type":...},
    "param-b" : { "haltype": "parameter", ...},
    ...
}
```

See `hal.query.param()` for description of the param dictionary details.

hal.query.signals() → dict

Retrieve information about all signals. The dictionary returned has the format:

```
{
    "sig-a" : { "haltype": "signal", "name": "sig-a", "type":...},
    "sig-b" : { "haltype": "signal", ...},
    ...
```

```
}
```

See `hal.query.signal()` for description of the signal dictionary details.

hal.query.comps() → dict

Retrieve information about all components. The dictionary returned has the format:

```
{
    "comp-a" : { "haltype": "component", "name": "comp-a", "type":...},
    "comp-b" : { "haltype": "component", ...},
    ...
}
```

See `hal.query.comp()` for description of the component dictionary details.

hal.query.functs() → dict

Retrieve information about all functions. The dictionary returned has the format:

```
{
    "funct-a" : { "haltype": "function", "name": "funct-a", ...},
    "funct-b" : { "haltype": "function", ...},
    ...
}
```

See `hal.query.funct()` for description of function dictionary details.

hal.query.threads() → dict

Retrieve information about all threads. The dictionary returned has the format:

```
{
    "thread-a" : { "haltype": "thread", "name": "thread-a", ...},
    "thread-b" : { "haltype": "thread", ...},
    ...
}
```

See `hal.query.thread()` for description of thread dictionary details.

13.3.6. Class `hal.shm`

The `shm` class is an interface to create and manage shared memory segments.

The `shm` class **should be considered *experimental*** and may or may not work as intended.

WARNING	Do not rely on this class. It may be removed in future releases.

shm methods

shm = hal.shm(*comp*:object, *key*:int, *size*:int)

Allocate a *size* sized shared memory segment with ID *key*.

shm.getbuffer()

Returns a Python `memoryview` object of the shared memory segment.

shm.setsize()

This is non-functional. **Do not use.** You cannot increase or decrease the shared memory segment's size once it is created.

13.3.7. HAL Port pipes

A HAL port is a byte-oriented pipe that streams bytes from the writer to the reader. It may be used to transport data between real-time and non-real-time in either direction. The HAL port pipe facility is distinct from the HAL stream facility in that it has no concept of typed data. The reader and writer of a port must handle a binary byte oriented data-stream. A HAL port uses HAL pins of type `hal.Type.PORT` to communicate.

Example - HAL port data writer:

```
import hal
import time
import struct

comp = hal.component("portwriter")

# Create the write-end
portw = comp.newpin("portpin", hal.Type.PORT, hal.Dir.OUT)

# Create a signal to link reader and writer
portsig = hal.new_sig("portsig", hal.Type.PORT)
portw.ready()

# Connect the pins to the signal net
hal.connect("portwriter.portpin", "portsig")
hal.connect("portreader.portpin", "portsig")

# Allocate and set the port's queue size
hal.set_s("portsig", 256)

while True:
    if portw.writable() < 2:
        time.sleep(0.001) # Don't busy-loop
        continue
    cmd, arg = read_command()
    binvals = struct.pack("bb", cmd, arg)
    portw.write(binvals)
```

Example - HAL port data reader component:

```
component portreader "Reads data from a HAL port";
pin in port portpin  "Port's read end";
```

```
description "Port reader example component";

option singleton;
option period no;
license "GPL";
function _;

;;
void do_abort(void) { /* you write code here */ }
void kill_switch(char x) { (void)x; /* you write some more code here */ }

FUNCTION(_)
{
    unsigned avail = hal_port_readable(portpin_ptr);
    if(avail > 1) {
        rtapi_u8 buf[2];
        if(!hal_port_read(portpin_ptr, buf, sizeof(buf))) {
            rtapi_print_msg(RTAPI_MSG_ERR, "Port read failed\n");
            hal_port_clear(portpin_ptr);  // Try to recover
        } else {
            switch(buf[0]) {
            case 'a': do_abort(); break;
            case 'k': kill_switch(buf[1]); break;
            // ...
            }
        }
    }
}
```

Example - HAL port hal file to load the reader:

```
loadrt portreader

addf portreader servo.thread
```

A more comprehensive implementation can be found in the *raster.comp* component together with the raster programmer.

IMPORTANT	A HAL port pipes can only have one (1) reader and one (1) writer. The port queue is not designed to support operation with multiple readers or writers.

NOTE	A HAL port queue cannot be allocated larger than 64 kiB (65536 bytes). The minimum size is one (1) byte, but that is not recommended. You should analyze your usage and find the appropriate size to set.

Port pin methods

port = comp.newpin(*name*:string, hal.Type.PORT, *io*:enum)

A port is a special pin type. It is created as a normal pin with type `hal.Type.PORT`. You need two pins for a port, one input and one output. Both ends of a port are connected with a signal (net). Setting the signal will allocate the port's queue.

port.readable() → int

Returns the number of bytes available for reading from the port queue.

port.writable() → int

Returns the number of bytes possible to write to the port queue.

port.write(*data*:bytes) → bool

Write a buffer onto the port queue. The argument *data* can be either a bytes buffer or a string. If it is a string, then it will be converted to a UTF-8 bytes buffer. Returns `True` if the *data* was successfully written to the port queue and `False` if not. The *write()* call fails if the port queue has not enough free capacity for the entire *data* argument.

NOTE	You should be careful when writing a string. Strings are Unicode encoded and may expand to multiple bytes when converted to UTF-8. Therefore, the length of the string may not match the length of the UTF-8 bytes buffer and not fit into the queue.

port.read(*size*:int) → bytes

Reads *size* bytes from the port and removes the bytes from the port queue. The port is tested whether *size* bytes are available before attempting to read. Returns a bytes buffer upon success or `False` on failure.

port.peek(*size*:int) → bytes

Reads *size* bytes from the port without removing the bytes from the port queue. The port is tested whether *size* bytes are available before attempting to peek. Returns a bytes buffer upon success or `False` on failure.

port.peek_commit(*size*:int) → bool

Removes *size* bytes from the port queue. Returns `True` on success or `False` if not.

Beispiel:

```
def wait_for(n):
    while port.readable() < n:
        time.sleep(0.01)

wait_for(2)
# Peek at the first 2 bytes of the data pipe
data = port.peek(2)
# Check the data for a pattern
if data[0] == 123 and data[1] == 42:
    port.peek_commit(2)  # Discard data from the pipe
else:
    # Not the discardable pattern, need 4 bytes then
    wait_for(4)
    data = port.read(4)
    handle_data(data)
```

port.clear()

Clears the content of the port queue.

port.size() → int

Return the allocated port queue size. The return value is zero (0) if no queue was allocated for the port.

13.4. Python module for managing the realtime backend

This documentation describes the `lcnc_realtime` Python module, which provides a Python API for managing the realtime backend.

This is a Python wrapper of the realtime script.

13.4.1. Grundlegende Verwendung (engl. basic usage)

Verify realtime capability and check if the realtime backend is running

```
import lcnc_realtime

print("lcnc_realtime.verify " + str(lcnc_realtime.verify()))
print("lcnc_realtime.verify_info " + str(lcnc_realtime.verify_info()))
print("lcnc_realtime.status " + str(lcnc_realtime.status()))
```

13.4.2. Methoden

lcnc_realtime.verify() → bool

Returns a boolean to indicate whether the system is realtime capable.

lcnc_realtime.verify_info() → tuple[bool, str]

Returns a tuple `(capable, type)`, where `capable` is the same boolean as `verify()` and `type` is a human-readable string naming the realtime type that rtapi is running (for example `Preempt RT`, `RTAI` or `Xenomai`), or `No realtime` when none is available.

lcnc_realtime.status() → bool

Returns a boolean to indicate whether the realtime backend is running.

13.5. GStat Python-Modul

13.5.1. Einführung

GStat ist eine Python-Klasse, die verwendet wird, um Nachrichten von LinuxCNC an andere Python-Programme zu senden. Sie verwendet GObject, um Nachrichten zu übermitteln, was es einfach macht, auf bestimmte Informationen zu hören. Dies wird als ereignisgesteuerte Programmierung bezeichnet, die effizienter ist als jedes Programm, das LinuxCNC zur gleichen Zeit abfragt. GladeVCP, Gscreen, Gmoccapy und QtVCP verwenden GStat ausgiebig. GStat befindet sich im Modul `hal_glib`.

Übersicht

- Zunächst importiert ein Programm das Modul "hal_glib" und instanziiert GStat.

- Dann stellt es eine Verbindung zu den Nachrichten her, die es überwachen möchte.
- GStat prüft den Status von LinuxCNC alle 100 ms und wenn es Unterschiede zur letzten Prüfung gibt, sendet es eine Callback-Nachricht an alle angeschlossenen Programme mit dem aktuellen Status.
- Wenn GStat die registrierte Funktion aufruft, sendet es das GStat-Objekt und alle Rückgabewerte der Nachricht.

Typische Codesignaturen:

```
GSTAT.connect('MESSGAE-TO-LISTEN-FOR', FUNCTION_TO_CALL)

def FUNCTION_TO_CALL(gstat_object, return_codes):
```

Oft wird LAMBDA verwendet, um das GSTAT-Objekt zu entfernen und die Rückgabecodes zu manipulieren:

```
GSTAT.connect('MESSGAE-TO-LISTEN-FOR', lambda o, return: FUNCTION_TO_CALL(not return))

def FUNCTION_TO_CALL(return_codes):
```

13.5.2. Beispiel für einen GStat-Code

Es gibt einige grundlegende Muster für die Verwendung von GStat, je nachdem, in welcher Bibliothek Sie sie verwenden. Wenn Sie GStat mit GladeVCP, Gscreen oder QtVCP verwenden, wird die GObject-Bibliothek nicht benötigt, da diese Toolkits GObject bereits einrichten.

Codemuster für HAL-Komponenten

Dieses Programm erzeugt zwei HAL-Pins, die den Status von G20/G21 ausgeben.

```
#!/usr/bin/env python3

import gi
gi.require_version('Gtk', '3.0')
from gi.repository import GObject
from gi.repository import GLib
import hal
from hal_glib import GStat
GSTAT = GStat()

# Callback zum Ändern des HAL-Pin-Status
def mode_changed(obj, data):
        h['g20'] = not data
        h['g21'] = data

# Make a component and pins
h = hal.component("metric_status")
h.newpin("g20", hal.Type.BOOL, hal.Dir.OUT)
h.newpin("g21", hal.Type.BOOL, hal.Dir.OUT)
h.ready()
```

```
# eine GSTAT-Nachricht mit einer Callback-Funktion verbinden
GSTAT.connect("metric-mode-changed",mode_changed)

# GSTAT zwingen, Zustände zu initialisieren
GSTAT.forced_update()

# loop till exit
try:
    GLib.MainLoop().run()
except KeyboardInterrupt:
    raise SystemExit
```

Dies würde mit `loadusr python PATH-TO-FILE/FILENAME.py` geladen werden oder wenn Sie warten müssen, bis die Pins hergestellt sind, bevor Sie fortfahren:
`loadusr python -Wn metric_status PATH-TO-FILE/FILENAME.py`
Die Pins lauten dann: `metric_status.g20` und `metric_status.g21`.

GladeVCP Python-Erweiterung Code-Muster

In dieser Datei wird davon ausgegangen, dass es drei GTK-Labels mit Namen gibt:

- *state_label*
- *e_state_label*
- *interp_state_label*

```
#!/usr/bin/env python3

from hal_glib import GStat
GSTAT = GStat()

class HandlerClass:

    def __init__(self, halcomp, builder, useropts):
        self.builder = builder

        GSTAT.connect("state-estop",lambda w: self.update_estate_label('ESTOP'))
        GSTAT.connect("state-estop-reset",lambda w: self.update_estate_label('RESET'))

        GSTAT.connect("state-on",lambda w: self.update_state_label('MACHINE ON'))
        GSTAT.connect("state-off",lambda w: self.update_state_label('MACHINE OFF'))

        GSTAT.connect("interp-paused",lambda w: self.update_interp_label('Paused'))
        GSTAT.connect("interp-run",lambda w: self.update_interp_label('Run'))
        GSTAT.connect("interp-idle",lambda w: self.update_interp_label('Idle'))

    def update_state_label(self,text):
        self.builder.get_object('state_label').set_label("State: %s" % (text))

    def update_estate_label(self,text):
        self.builder.get_object('e_state_label').set_label("E State: %s" % (text))

    def update_interp_label(self,text):
        self.builder.get_object('interp_state_label').set_label("Interpreter State: %s" %
```

```
(text))

def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,builder,useropts)]
```

QtVCP Python-Erweiterungscode-Muster

QtVCP erweitert GStat, muss also anders geladen werden, aber alle Meldungen sind in QtVCP verfügbar. Diese Handler-Datei geht davon aus, dass es drei QLabels mit Namen gibt:

- *state_label*
- *e_state_label*
- *interp_state_label*

```
#!/usr/bin/env python3

from qtvcp.core import Status
GSTAT = Status()

class HandlerClass:

    def __init__(self, halcomp,widgets,paths):
        self.w = widgets

        GSTAT.connect("state-estop",lambda w: self.update_estate_label('ESTOP'))
        GSTAT.connect("state-estop-reset",lambda w: self.update_estate_label('RESET'))

        GSTAT.connect("state-on",lambda w: self.update_state_label('MACHINE ON'))
        GSTAT.connect("state-off",lambda w: self.update_state_label('MACHINE OFF'))

        GSTAT.connect("interp-paused",lambda w: self.update_interp_label('Paused'))
        GSTAT.connect("interp-run",lambda w: self.update_interp_label('Run'))
        GSTAT.connect("interp-idle",lambda w: self.update_interp_label('Idle'))

    def update_state_label(self,text):
        self.w.state_label.setText("State: %s" % (text))

    def update_estate_label(self,text):
        self.w.e_state_label.setText("E State: %s" % (text))

    def update_interp_label(self,text):
        self.winterp_state_label.setText("Interpreter State: %s" % (text))

def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,widgets,paths)]
```

13.5.3. Nachrichten

periodic (engl. für periodisch)

(gibt nichts zurück) - wird alle 100 ms gesendet.

state-estop

(gibt nichts zurück) - Wird gesendet, wenn LinuxCNC in den Notaus geht.

state-estop-reset

(gibt nichts zurück) - Wird gesendet, wenn LinuxCNC aus dem Notaus heraus kommt.

state-on

(gibt nichts zurück) - Wird gesendet, wenn LinuxCNC im Zustand Maschine ein ist.

state-off

(gibt nichts zurück) - Wird gesendet, wenn sich LinuxCNC im ausgeschalteten Zustand befindet.

homed

(returns string) - Wird gesendet, wenn jeder Joint in die Ausgangsposition gebracht wird.

all-homed

(gibt nichts zurück) - Wird gesendet, wenn alle definierten Gelenke referenziert sind.

not-all-homed

(gibt eine Zeichenkette zurück) - Sendet eine Liste von Gelenken, die derzeit nicht referenziert sind.

override_limits_changed

(returns string) - Wird gesendet, wenn LinuxCNC angewiesen wurde, seine Grenzen zu überschreiten.

hard-limits-tripped

(liefert bool, Python-Liste) - Wird gesendet, wenn ein hartes Limit ausgelöst wird. bool zeigt an, dass ein Limit ausgelöst wurde, die Liste zeigt alle aktuellen Grenzwerte der verfügbaren Gelenke.

mode-manual

(returns nothing) - Wird gesendet, wenn LinuxCNC in den manuellen Modus wechselt.

mode-mdi

(returns nothing) - Wird gesendet, wenn LinuxCNC in den MDI-Modus wechselt.

mode-auto

(gibt nichts zurück) - Wird gesendet, wenn LinuxCNC in den Auto-Modus wechselt.

command-running

(gibt nichts zurück) - Wird gesendet, wenn ein Programm oder MDI ausgeführt wird

command-stopped

(gibt nichts zurück) - Wird gesendet, wenn ein Programm oder MDI angehalten wurde

command-error

(returns nothing) - Wird gesendet, wenn bei der Ausführung eines Befehls ein Fehler auftritt

interp-run

(gibt nichts zurück) - Wird gesendet, wenn der Interpreter von LinuxCNC ein MDI oder Programm ausführt.

interp-idle

(returns nothing) - Wird gesendet, wenn der Interpreter von LinuxCNC im Leerlauf (engl. idle) ist.

interp-paused

(gibt nichts zurück) - Wird gesendet, wenn der Interpreter von LinuxCNC pausiert ist.

interp-reading

(gibt nichts zurück) - Wird gesendet, wenn der Interpreter von LinuxCNC gerade liest.

interp-waiting

(gibt nichts zurück) - Wird gesendet, wenn der Interpreter von LinuxCNC wartet.

jograte-changed

(gibt Float zurück) - Wird gesendet, wenn sich die Jog-Rate geändert hat.
LinuxCNC verfügt nicht über eine interne Jog-Geschwindigkeit.
Dies ist die interne Jog-Rate von GStat.
Es wird erwartet, dass sie in den nativen Einheiten der Maschine angegeben wird, unabhängig vom aktuellen Einheitenmodus.

jograte-angular-changed

(gibt Float zurück) - Wird gesendet, wenn sich die Jog-Rate geändert hat.
LinuxCNC verfügt über keine interne Jog-Geschwindigkeit.
Dies ist die interne Jog-Rate von GStat.
Es wird erwartet, dass sie in den nativen Einheiten der Maschine angegeben wird, unabhängig vom aktuellen Einheitenmodus.

jogincrement-changed

(liefert Float, Text) - Wird gesendet, wenn sich die Schrittweite geändert hat.
LinuxCNC verfügt über keine interne Schrittweite.
Dies ist die interne Schrittweite von GStat.
Es wird erwartet, dass es in den nativen Einheiten der Maschine angegeben wird, unabhängig vom aktuellen Einheitenmodus.

jogincrement-angular-changed

(gibt float, text zurück) - Wird gesendet, wenn sich das Winkeljog-Inkrement geändert hat.
LinuxCNC hat kein internes Winkeljog-Inkrement.
Dies ist das interne Winkeljog-Inkrement von GStat.
Es wird erwartet, dass es sich unabhängig vom aktuellen Einheitenmodus in den nativen Einheiten der Maschine befindet.

program-pause-changed

(gibt bool zurück) - Wird gesendet, wenn das Programm pausiert/unpausiert wird.

optional-stop-changed

(gibt bool zurück) - Wird gesendet, wenn der optionale Stopp gesetzt/entfernt wird

block-delete-changed

(gibt bool zurück) - wird gesendet, wenn der Block delete gesetzt/gelöscht wird.

file-loaded

(liefert String) - Wird gesendet, wenn LinuxCNC eine Datei geladen hat

reload-display

(gibt nichts zurück) - Wird gesendet, wenn eine Anforderung zum Neuladen der Anzeige vorliegt

line-changed

(gibt eine ganze Zahl zurück) - Wird gesendet, wenn LinuxCNC eine neue Zeile gelesen hat.
LinuxCNC aktualisiert diese nicht für jede Art von Zeile.

tool-in-spindle-changed

(gibt ganze Zahl zurück) - Wird gesendet, wenn sich das Werkzeug geändert hat.

tool-info-changed

(gibt Python-Objekt zurück) - Wird gesendet, wenn sich die aktuelle Werkzeuginformation ändert.

current-tool-offset

(gibt Python-Objekt zurück) - Wird gesendet, wenn sich der aktuelle Werkzeugversatz ändert.

motion-mode-changed

(gibt ganze Zahl zurück) - Wird gesendet, wenn sich der Modus der Bewegung geändert hat

spindle-control-changed

(gibt Integer, Bool, Integer, Bool zurück) - (Spindelnummer, Zustand der eingeschalteten Spindel, angeforderte Spindelrichtung und -geschwindigkeit, Zustand bei Drehzahl)
Wird gesendet, wenn sich die Spindelrichtung oder der Laufstatus ändert oder die Drehzahl sich ändert.

current-feed-rate

(liefert Float) - Wird gesendet, wenn sich die aktuelle Vorschubgeschwindigkeit ändert.

current-x-rel-position

(returns float) - Wird alle 100 ms gesendet.

current-position

(liefert pyobject, pyobject, pyobject, pyobject) - Wird alle 100 ms gesendet.
liefert Tupel von Position, relativer Position, Restweg und der aktuellen Gelenkposition. Vor der Referenzfahrt ist bei Multigelenkachsen nur die Gelenkposition gültig.

current-z-rotation

(liefert Float) - Wird gesendet, wenn sich der aktuelle Drehwinkel um die Z-Achse ändert

requested-spindle-speed-changed

(liefert Float) - Wird gesendet, wenn sich die aktuell angeforderte Drehzahl ändert

actual-spindle-speed-changed

(returns float) - Wird gesendet, wenn sich die tatsächliche Drehzahl basierend auf dem HAL-Pin `spindle.0.speed-in` ändert.

spindle-override-changed

(returns float) - Wird gesendet, wenn sich der Spindel-Override-Wert ändert
in Prozent

feed-override-changed

(returns float) – Wird gesendet, wenn sich der Feed-Override-Wert ändert
in Prozent

rapid-override-changed

(returns float) – Wird gesendet, wenn sich der Rapid-Override-Wert ändert
in Prozent (0-100)

max-velocity-override-changed

(returns float) - Wird gesendet, wenn sich der Override-Wert der maximalen Geschwindigkeit ändert
in Einheiten pro Minute

feed-hold-enabled-changed

(returns bool) - Wird gesendet, wenn sich der Feed-Hold-Status ändert

itime-mode

(gibt bool zurück) - Wird gesendet, wenn sich der G93-Status ändert
(inverser Zeitmodus)

fpm-mode

(returns bool) - Wird gesendet, wenn sich der G94-Status ändert
(Vorschub pro Minute Modus)

fpr-mode

(returns bool) - Wird gesendet, wenn sich der G95-Status ändert
(Vorschub pro Umdrehung Modus)

css-mode

(liefert bool) - Wird gesendet, wenn sich der G96-Status ändert +
(Modus konstanter Oberflächenvorschub, engl. constant surface feed mode)

rpm-mode

(returns bool) - Wird gesendet, wenn sich der G97-Status ändert
(Modus mit konstanter Drehzahl, engl. constant RPM mode)

radius-mode

(gibt bool zurück) - Wird gesendet, wenn sich der G8-Status ändert +
Anzeige X in Radian - Modus

diameter-mode

(gibt bool zurück) - Wird gesendet, wenn sich der G7-Status ändert
Anzeige X im Diameter-Modus

flood-changed

(gibt boolean zurück) - Wird gesendet, wenn sich der Flutkühlmittelzustand ändert.

mist-changed

(liefert boolean) - Wird gesendet, wenn sich der Zustand des Nebelkühlmittels ändert.

m-code-changed

(gibt String zurück) - Wird gesendet, wenn sich aktive M-Codes ändern

g-code-changed

(gibt String zurück) - Wird gesendet, wenn sich der aktive G-Code ändert

metric-mode-changed

(gibt bool zurück) - Wird gesendet, wenn sich der G21-Status ändert

user-system-changed

(gibt String zurück) - Wird gesendet, wenn sich das Referenzkoordinatensystem (G5x) ändert

mdi-line-selected

(returns string, string) - soll gesendet werden, wenn eine MDI-Zeile vom Benutzer ausgewählt wird. +
Dies hängt von den verwendeten Widgets / Bibliotheken ab.

gcode-line-selected

(gibt eine ganze Zahl zurück) - soll gesendet werden, wenn eine G-Code-Zeile vom Benutzer ausgewählt wird. +
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

graphics-line-selected

(gibt eine ganze Zahl zurück) - soll gesendet werden, wenn der Benutzer eine Grafikzeile auswählt. +
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

graphics-loading-progress

(gibt eine ganze Zahl zurück) - soll den Prozentsatz zurückgeben, der beim Laden eines Programms oder bei der Ausführung eines Programms erreicht wurde.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

graphics-gcode-error

(gibt String zurück) - soll gesendet werden, wenn beim Laden ein G-Code-Fehler gefunden wird.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

graphics-gcode-properties

(gibt ein Python-dict zurück) - gibt eine benamste Liste aller Gelenke zurück, die aktuell einem Fehler folgen.
Das dict enthält Einträge zu den folgenden Namen (engl. keys):

- `name` (*string*): Name der geladenen Datei
- `size` (*string*): Größe in Bytes und Zeilen
- `g0` (*string*): Gesamtschnelldistanz
- `g1` (*string*): Gesamtvorschubweg
- `run` (*string*): Geschätzte Programmlaufzeit
- `toollist` (*list*): Liste der verwendeten Werkzeuge
- `x` (*string*): X Ausmaße (Grenzen) [1]
- `x_zero_rxy` (*string*): X-Ausdehnungen ohne Drehung um z (Grenzen) [1]
- `y` (*string*): Y-Ausdehnungen (Grenzen) [1]
- `y_zero_rxy` (*string*): Y-Ausdehnungen ohne Drehung um Z (Grenzen) [1]
- `z` (*string*): Z-Ausdehnungen (Grenzen) [1]
- `z_zero_rxy` (*string*): Z-Ausdehnungen ohne Drehung um z (Grenzen) [1]
- `machine_unit_sys` (*string*): Maschineneinheiten (*Metrisch* oder *Imperial*)
- `gcode_units` (*string*): Einheiten in G-Code-Datei (*mm* oder *in*)

NOTE

1. Siehe die Bilder extends non-rotated und extends rotated 30 degrees zum besseren Verständnis.

graphics-program-time

(returns rs274.program_time.ProgramTime) - Sent when G-code is loaded, right after **graphics-gcode-properties**.
The program's time estimate as numbers rather than as prose: `total` (seconds at 100% override, or `None` when the machine's INI states no velocity limits and the program could not be timed), the `rapid_time`/`feed_time`/`stop_time` shares of it, `flags` naming what could not be modelled, and the lookups `at_line(line)`, `fraction(line)` and `remaining(line)` over the parse's `[line, cumulative seconds]` table. Each takes an optional `elapsed_hint`, which picks between the occurrences of a line that a loop runs more than once. The same object is available from the graphics widget as `get_program_time()`.

graphics-view-changed

(liefert string, Python dict oder None) - soll gesendet werden, wenn die Grafikansicht geändert wird.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

mdi-history-changed

(gibt keine zurück) - soll gesendet werden, wenn ein MDI-Verlauf neu geladen werden muss.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

machine-log-changed

(gibt keine zurück) - soll gesendet werden, wenn sich das Maschinenprotokoll geändert hat.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

update-machine-log

(gibt String, String zurück) - soll bei der Aktualisierung der Maschine gesendet werden.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

move-text-lineup

(gibt None zurück) - soll gesendet werden, wenn der Cursor in der G-Code-Anzeige um eine Zeile nach oben bewegt wird.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

move-text-linedown

(gibt None zurück) - soll gesendet werden, wenn der Cursor in der G-Code-Anzeige eine Zeile nach unten bewegt wird.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

dialog-request

(gibt Python dict zurück) - soll gesendet werden, wenn ein GUI-Dialog angefordert wird.
Es verwendet ein Python dict für die Kommunikation. Das dict muss das folgende Keyname-Paar enthalten:

- NAME: *angeforderter Dialogname*
  Das dict (engl. kurz für dictionary, eine Python Funktionalität) hat normalerweise mehrere Keyname-Paare - das hängt vom Dialog ab.
  Dialoge geben Informationen über eine allgemeine Nachricht zurück
  Dies hängt von den verwendeten Widgets/Bibliotheken ab.

focus-overlay-changed

(gibt bool, string, Python object) - soll gesendet werden, wenn ein Overlay über die Anzeige gelegt werden soll.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

play-sound

(returns string) - soll gesendet werden, wenn eine bestimmte Sounddatei angefordert wird, die abgespielt werden soll.
Dies hängt von den verwendeten Widgets / Bibliotheken ab.

virtual-keyboard

(gibt String zurück) - soll gesendet werden, wenn eine Bildschirmtastatur angefordert wird.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

dro-reference-change-request

(liefert eine ganze Zahl) - soll gesendet werden, wenn ein DRO-Widget aufgefordert wird, seine Referenz zu ändern.
0 = Maschine, 1 = relativ, 3 = Restweg
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

show-preferences

(returns string) - soll gesendet werden, wenn eine bestimmte Sounddatei angefordert wird, die abgespielt werden soll.
Dies hängt von den verwendeten Widgets / Bibliotheken ab.

shutdown

(gibt None zurück) - soll gesendet werden, wenn LinuxCNC zum Herunterfahren aufgefordert wird.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

status-message (engl. für Status-Meldung)

liefert Python-Dict (message), Python-Dict (options) Vorgesehen für einen Bildschirm/Panel, um Status- oder Protokollmeldungen von Widgets abzurufen, kann aber allgemein verwendet werden.
Das hörende Objekt sollte mindestens nach den folgenden Einträgen suchen und diese verarbeiten:

Das message dict würde beinhalten:

- TITLE: (string)
- SHORTTEXT: (string)
- DETAILS: (string)

Das options dict würde beinhalten:

- LEVEL: (integer)
- LOG: (bool)

Das lauschende Objekt kann dies verwenden, um Informationen in einer Textzeile oder einem Meldungsdialog anzuzeigen.
LEVEL gibt die Dringlichkeit an: 0 = STANDARD, 1 = WARNUNG, 2 = KRITISCH
LOG gibt an, ob die Meldung – falls verfügbar – in eine Datei/Seite protokolliert werden soll.
Es wird angenommen, dass LOG-Meldungen den DETAILS-Eintrag verwenden.

Ein Beispiel für das Versenden einer Bedienermeldung:

```
mess = {'SHORTTEXT':'File Copy Failed',
        'TITLE':'FileManager',
        'DETAILS':'Es ist nicht genug Platz auf der Platte, um die Datei zu kopieren'}
opt = {'LOG':True,'LEVEL':2}
STATUS.emit('status-message',mess,opt)
```

Ein Beispiel für ein empfangendes (engl. listening) Objektes:

```
# STATUS mitteilen was wir antworten möchten zu irgendwelchen gesendeten 'status-message'
Nachrichten.
STATUS.connect('status-message', lambda w, d, o: self.add_external_status(m,o))

    def add_external_status(self, message, option):

        # Extrahiere und fange (engl. trap) Fehler für erwartete Einträge
        level = option.get('LEVEL', STATUS.DEFAULT)
```

```
        log = option.get("LOG", True)
        title = message.get('TITLE', '')
        mess = message.get('SHORTTEXT', '')
        logtext = message.get('DETAILS', '')

        # Rufe Funktion auf zur Anzage der Nachricht auf der Zustands-Anzeige (entl.
status bar):
        self.add_status(mess, level, noLog=True)

        # Verlange Update der Log-Datei
        if log:
            STATUS.emit('update-machine-log', "{}\n{}".format(title, logtext), 'TIME')
```

error

(liefert Integer, String) - soll gesendet werden, wenn ein Fehler gemeldet wurde.
integer steht für die Art des Fehlers. ERROR, TEXT oder DISPLAY
string ist die eigentliche Fehlermeldung.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

general

(gibt Python dict zurück) - soll gesendet werden, wenn eine Nachricht gesendet werden muss, die nicht von einer spezifischeren Nachricht abgedeckt wird.
Die allgemeine Nachricht sollte spärlich als vernünftig verwendet werden, da alle damit verbundenen Objekte sie analysieren müssen.
Es verwendet ein Python-Diktum für die Kommunikation.
Das Diktat sollte ein eindeutiges ID-Schlüsselnamenpaar enthalten und darauf überprüft werden:

- Kennung: *UNIQUE_ID_CODE*
  Das Dikt hat normalerweise mehr Schlüsselnamenpaar - es hängt von der Implementierung ab.

erzwungene Aktualisierung (engl. forced update)

(gibt keine zurück)" - soll gesendet werden, wenn ein Objekt initialisiert oder willkürlich aktualisiert werden soll.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

Fortschritt (engl. progress)

(liefert Integer, Python-Objekt) - soll gesendet werden, um den Fortschritt eines Filterprogramms anzuzeigen.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

Folgefehler

(gibt eine Python-Liste zurück) - gibt eine Liste aller Gelenke zurück, die aktuell einem Fehler folgen.

13.5.4. Funktionen

Dies sind Komfortfunktionen, die in der Programmierung häufig verwendet werden.

set_jograte

(float) - LinuxCNC hat kein internes Konzept für die Jog-Geschwindigkeit (engl. jog rate) - jede GUI hat

ihre eigene. Dies ist nicht immer bequem.
Diese Funktion ermöglicht es, eine Tipprate für alle Objekte einzustellen, die mit dem Signal `jograte-changed` verbunden sind.
Der Standardwert ist 15.
GSTAT.set_jog_rate(10) würde die Jog-Rate auf 10 Maschineneinheiten pro Minute setzen und das `jograte-changed`-Signal ausgeben.

get_jograte()

(Nothing) - x = GSTAT.get_jograte() würde die aktuelle interne Jograte von GSTAT (float) zurückgeben.

set_jograte_angular

(float) -

get_jograte_angular

(None) -

set_jog_increment_angular

(float, string) -

get_jog_increment_angular

(None) -

set_jog_increments

(float, string) -

get_jog_increments

(None) -

ist_all_homed

(nothing) - Dies gibt den aktuellen Status von all_homed (BOOL) zurück.

machine_is_on

(nothing) - Dies gibt den aktuellen Zustand der Maschine zurück (BOOL).

estop_is_clear

(nichts) - Gibt den Status von Estop zurück (BOOL)

set_tool_touchoff

(Werkzeug,Achse,Wert) - Dieser Befehl wird

1. den aktuellen Modus aufzeichnen,
2. Wechsel zu MDI-Modus,
3. den MDI-Befehl aufrufen: G10 L10 P[WERKZEUG] [ACHSE] [WERT],
4. warten, bis der Vorgang abgeschlossen ist,
5. G43 aufrufen,
6. warten, bis der Vorgang abgeschlossen ist,

7. zurück in den ursprünglichen Modus wechseln.

set_axis_origin

(Achse,Wert) - Dieser Befehl wird

1. den aktuellen Modus aufzeichnen,
2. Wechsel zu MDI-Modus,
3. den MDI-Befehl aufzurufen: G10 L20 P0 [ACHSE] [WERT],
4. warten, bis der Vorgang abgeschlossen ist,
5. zurück in den ursprünglichen Modus wechseln,
6. ein *Reload-Display*-Signal aussenden.

do_jog

(Achsennummer, Richtung, Abstand) - Diese Funktion bewegt eine Achse kontinuierlich oder in einem bestimmten Abstand.
Sie müssen sich im richtigen Modus befinden, um zu joggen.

check_for_modes

(mode) - Diese Funktion prüft, ob der LinuxCNC-Modus erforderlich ist.
Es gibt ein Python-Tupel zurück (Status, Modus)
-Modus wird in dem Modus eingestellt, in dem sich das System befindet
wird wie folgt festgelegt:

- false, wenn mode 0 ist
- false, wenn die Maschine beschäftigt ist
- wahr, wenn LinuxCNC im angeforderten Modus ist
- Keine, wenn eine Änderung möglich ist, aber nicht im gewünschten Modus

get_current_mode

(nothing) - gibt eine ganze Zahl zurück: den aktuellen LinuxCNC-Modus.

set_selected_joint

(Integer) - speichert intern die ausgewählte Gelenknummer.
fordert die Auswahl des Gelenks an, indem er die Meldung
Nachricht "Gelenkauswahl geändert".

get_selected_joint

(None) - gibt eine ganze Zahl zurück, welche die interne ausgewählte Gelenknummer darstellt.

set_selected_axis

(string) - speichert intern den ausgewählten Achsenbuchstaben.
Fordert die Auswahl der Achse durch die `axis-selection-changed`-Meldung (engl. für "Achsenauswahl-geändert") an.

get_selected_axis

(None) - gibt eine Zeichenkette zurück, die den intern ausgewählten Achsenbuchstaben darstellt.

is_man_mode

(None) -

is_mdi_mode

(None) -

is_auto_mode

(None) -

is_on_and_idle

(None) -

is_auto_running

(None) -

is_auto_paused

(None) -

is_file_loaded

(None) -

is_metric_mode

(None) -

is_spindle_on

(None) -

shutdown

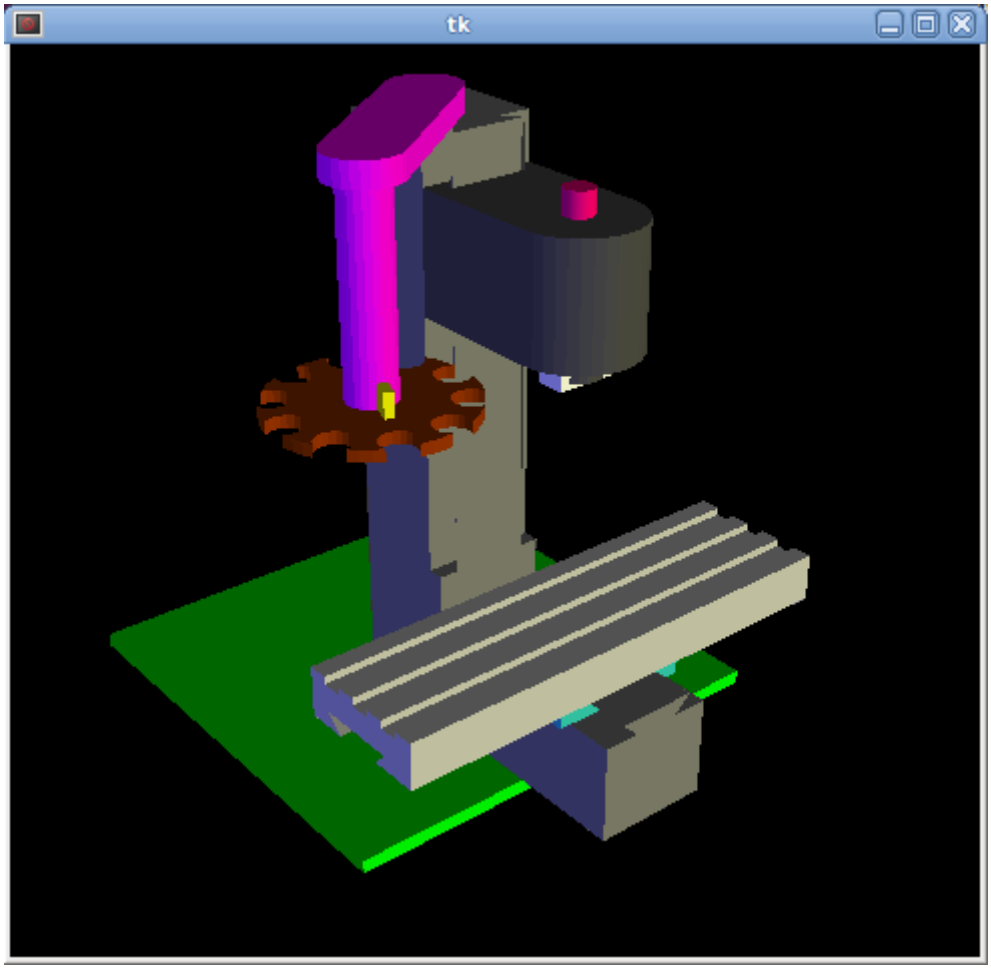
(None) -

13.5.5. Bekannte Probleme

Einige Statuspunkte werden während eines laufenden Programms falsch gemeldet, da der Interpreter der aktuellen Position eines laufenden Programms vorausläuft. Dies wird hoffentlich mit der Integration der State-Tags-Entwicklungs-Zweiges behoben.

13.6. Vismach

Vismach ist eine Reihe von Python-Funktionen, mit denen Modelle von Maschinen erstellt und animiert werden können. Vismach zeigt das Modell in einem 3D-Ansichtsfenster an und die Modellteile werden animiert, wenn sich die Werte der zugehörigen HAL-Pins ändern.



Das Vismach Ansichtsfenster (engl. viewport view) kann wie folgt manipuliert werden:

- **Zoom** durch Scrollrad oder Ziehen mit der rechten Taste,
- **Schwenk** durch Ziehen der linken Taste,
- **Drehen** durch Ziehen mit der mittleren Taste oder durch Ziehen mit der Umschalttaste.

Ein Vismach-Modell hat die Form eines Python-Skripts und kann die Standard-Python-Syntax verwenden. Das bedeutet, dass es mehr als eine Möglichkeit gibt, das Skript zu erstellen, aber in den Beispielen in diesem Dokument werde ich die einfachste und grundlegendste davon verwenden.

Die grundlegende Reihenfolge bei der Erstellung des Vismach-Modells ist

- Erstellen von HAL-Pins zur Steuerung der Bewegung.
- Erstellen der Werkstücke (engl. parts).
- Definiere, wie sie sich bewegen.
- In Bewegungsgruppen zusammenstellen.

13.6.1. Starten des Skripts

Zum Testen ist es nützlich, die *#!/usr/bin/env python3* anzugeben, damit die Datei als Skript ausgeführt werden kann. Als Erstes müssen die erforderlichen Bibliotheken importiert werden.

```
#!/usr/bin/env python3

from vismach import *
import hal
import math
import sys
```

13.6.2. Erstellen der HAL-Pins.

HAL-Pins werden mit der normalen Python-Bibliothek "hal" erstellt und sind nicht spezifisch für Vismach. Weitere Einzelheiten finden Sie im Abschnitt Erstellen von Nicht-Echtzeit Komponenten in Python. Es sollte eine Komponente mit einem Namen erstellt werden, der mit dem Namen der Skriptdatei übereinstimmt, und dann werden die HAL-Pins zu dieser Komponente hinzugefügt. Sie werden mit ihrem Komponenten-Handle und Kurznamen referenziert, wenn sie zur Animation des Vismach-Modells verwendet werden.

```
c = hal.component("samplegui")
c.newpin("joint0", hal.Type.REAL, hal.Dir.IN)
c.newpin("joint1", hal.Type.REAL, hal.Dir.IN)
c.ready()
```

Erzeugt die HAL-Pins *samplegui.joint0* und *samplegui.joint1*. Beim Laden des Vismach-Modells mit *loadusr -W samplegui* teilt die Funktion *c.ready()* loadusr mit, dass es bereit ist.

13.6.3. Erstellen von Teilen

Am einfachsten ist es wahrscheinlich, die Geometrie in einem CAD-Paket zu erstellen und mit den Funktionen `AsciiSTL()` oder `AsciiOBJ()` in das Modellskript zu **importieren**. Beide Funktionen können eines von zwei benannten Argumenten annehmen, entweder einen Dateinamen oder Rohdaten:

- Teil = AsciiSTL(Dateiname="path/to/file.stl")
  Teil = AsciiSTL(data="solid part1 facet normal ....")
  Teil = AsciiOBJ(Dateiname="Pfad/Zu/Datei.obj")
  Teil = AsciiOBJ(data="v 0.123 0.234 0.345 1.0 ...") `
  Die Teile werden im Vismach-Raum an denselben Stellen erstellt, an denen sie sich im STL- oder OBJ-Raum befinden. Das bedeutet, dass es möglich sein kann, das Modell im CAD-Paket zusammenzusetzen.

Alternativ können Teile innerhalb des Modellskripts aus einer Reihe von **Form-Primitiven** erstellt werden. Viele Formen werden am Ursprung erstellt und müssen nach der Erstellung an die gewünschte Stelle verschoben werden:

- `cylinder = CylinderX(x1, r1, x2, r2) + cylinder = CylinderY(y1, r1, y2, r2) + cylinder = CylinderZ(z1, r1, z2, r2)`
  Erzeugt einen (optional verjüngten) Zylinder auf der angegebenen Achse mit den angegebenen Radien an den angegebenen Punkten auf der Achse.
- sphere = Sphere(x, y, z, r) `
  Erzeugt eine Kugel mit Radius r bei (x,y,z)

- `triangle = TriangleXY(x1, y1, x2, y2, x3, y3, z1, z2) + triangle = TriangleXZ(x1, z1, x2, z2, x3, z3, y1, y2) + triangle = TriangleYZ(y1, z1, y2, z2, y3, z3, x1, x2)`
  Erzeugt eine dreieckige Platte zwischen Ebenen, die durch die letzten beiden Werte parallel zur angegebenen Ebene definiert sind, wobei die Eckpunkte durch die drei Koordinatenpaare gegeben sind.
- `Bogen = ArcX(x1, x2, r1, r2, a1, a2)`
  Erstellt eine Bogenform.
- `box = Box(x1, y1, z1, x2, y2, z2)`
  Erzeugt ein rechteckiges Prisma mit gegenüberliegenden Ecken an den angegebenen Positionen und Kanten parallel zu den XYZ-Achsen.
- `box = BoxCentered(xw, yw, zw)`
  Erzeugt eine xw mal yw mal zw Box, die auf den Ursprung zentriert ist.
- `box = BoxCenteredXY(xw, yw, z)`
  Erzeugt eine Box mit der Breite xw / yw und der Höhe z.

Zusammengesetzte Teile können durch **Zusammensetzen** dieser Primitive entweder zur Erstellungszeit oder nachträglich mittels `Collection()` erstellt werden:

```
part1 = Collection([Sphere(100,100,100,50), CylinderX(100,40,150,30)])
part2 = Box(50,40,75,100,75,100)
part3 = Collection([part2, TriangleXY(10,10,20,10,15,20,100,101)])
part4 = Collection([part1, part2])
```

13.6.4. Bewegliche Teile

Teile müssen möglicherweise im Vismach-Raum verschoben werden, um das Modell zusammenzusetzen. Sie müssen möglicherweise auch verschoben werden, um die Animation zu erstellen, da die Rotationsachse der Animation am Ursprung erstellt wird (sich aber mit dem Teil bewegt):

- `Teil1 = Translate([Teil1], x, y, z)`
  Verschiebe Teil1 um die angegebenen Abstände in x, y und z.
- Teil1 = Rotate([Teil1], theta, x, y, z)`
  Dreht das Teil um den Winkel theta um eine Achse zwischen dem Ursprung und x, y, z.

13.6.5. Animieren von Teilen

Um das Modell zu animieren (gesteuert durch die Werte der HAL-Pins) gibt es die beiden Funktionen "HalTranslate" und "HalRotate". Damit sich Teile in einer Baugruppe bewegen können, müssen ihre HAL-Bewegungen definiert werden, bevor sie mit dem Befehl "Collection" zusammengesetzt werden. Die Rotationsachse und der Translationsvektor bewegen sich mit dem Teil, wenn es vom Vismach-Skript während des Zusammenbaus des Modells bewegt wird, oder wenn es sich als Reaktion auf die HAL-Pins bewegt, während das Modell animiert wird:

- `part = HalTranslate([part], comp, "hal_pin", xs, ys, zs)`

Die Funktionsargumente sind:

- zuerst eine *Sammlung/ein Teil*, die vorher im Skript erstellt werden kann, oder an dieser Stelle erstellt werden kann, wenn dies bevorzugt wird, zB `part1 = HalTranslate([Box(....)], ...)`.
- Die *HAL-Komponente* ist das nächste Argument, d.h. das Objekt, das durch den Befehl `comp = hal.component(...)` zurückgegeben wird. Danach folgt der Name der HAL-Komponente, welche die Bewegung animieren soll. Dieser muss mit einem bestehenden HAL-Pin übereinstimmen, der Teil der zuvor im Skript erstellten HAL-Komponente ist.
- Dann folgen Sie den *X, Y, Z-Skalen*.
  Bei einer kartesischen Maschine, die im Maßstab 1:1 erstellt wurde, wäre dies normalerweise 1,0,0 für eine Bewegung in positiver X-Richtung.
  Wenn die STL-Datei jedoch in cm und die Maschine in Zoll erstellt wurde, kann dies an dieser Stelle durch die Verwendung von 0,3937 (1cm /2.54in) als Maßstab korrigiert werden.

- `part = HalRotate([part], comp, "hal_pin", angle_scale, x, y, z)`
  Dieser Befehl ähnelt in seiner Funktionsweise HalTranslate, außer dass es normalerweise notwendig ist, das Teil zuerst zum Ursprung zu bewegen, um die Achse zu definieren.
  - Die *Drehachse* verläuft vom Ursprungspunkt zu dem durch (x,y,z) definierten Punkt.
    Wenn das Teil vom Ursprung an seinen richtigen Ort zurückbewegt wird, kann man davon ausgehen, dass die Drehachse im Teil "eingebettet" bleibt.
  - Drehwinkel werden in Grad angegeben. Für ein Drehgelenk mit einer Skalierung von 0-1 müssten Sie also eine Winkelskala von 360 verwenden.

13.6.6. Zusammenbau des Modells.

Damit sich die Teile gemeinsam bewegen können, müssen sie mit dem Befehl Collection() zusammengefügt werden. Es ist wichtig, die Teile zusammenzufügen und ihre Bewegungen in der richtigen Reihenfolge zu definieren. Um zum Beispiel eine Fräsmaschine mit beweglichem Kopf, einer rotierenden Spindel und einer animierten Zugstange zu erstellen, würden Sie dies tun:

- Erstellen Sie den Hauptteil des Kopfes.
- Erstellen Sie die Spindel im Ursprung.
- Definieren Sie die Drehung.
- Bewegen Sie den Kopf zur Spindel oder die Spindel zum Kopf.
- Erstellen Sie die Zugstange (engl. draw bar).
- Definieren Sie die Bewegung der Zugstange.
- Bauen Sie die drei Teile zu einer Kopfeinheit zusammen.
- Definieren Sie die Bewegung der Kopfeinheit.

In diesem Beispiel wird die Spindeldrehung durch die Drehung eines Satzes von Mitnehmern angezeigt:

```
#Drive dogs
dogs = Box(-6,-3,94,6,3,100)
dogs = Color([1,1,1,1],[dogs])
```

```
dogs = HalRotate([dogs],c,"spindle",360,0,0,1)
dogs = Translate([dogs],-1,49,0)

#Drawbar
draw = CylinderZ(120,3,125,3)
draw = Color([1,0,.5,1],[draw])
draw = Translate([draw],-1,49,0)
draw = HalTranslate([draw],c,"drawbar",0,0,1)

# head/spindle
head = AsciiSTL(filename="./head.stl")
head = Color([0.3,0.3,0.3,1],[head])
head = Translate([head],0,0,4)
head = Collection([head, tool, dogs, draw])
head = HalTranslate([head],c,"Z",0,0,0.1)

# base
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# mount head on it
base = Collection([head, base])
```

Schließlich muss eine einzige Sammlung aller Maschinenteile, Böden und Arbeiten (falls vorhanden) erstellt werden:

- Für eine *serial machine* wird jedes neue Teil der Sammlung des vorherigen Teils hinzugefügt.
- Bei einer *Parallelmaschine* kann es mehrere "Basis"-Teile geben.

So wird zum Beispiel in scaragui.py link3 zu link2, link2 zu link1 und link1 zu link0 hinzugefügt, so dass das endgültige Modell wie folgt erstellt wird:

```
model = Collection([link0, floor, table])
```

Bei einem VMC-Modell mit separaten Teilen, die sich auf dem Sockel bewegen, könnte dies der Fall sein:

```
model = Collection([base, saddle, head, carousel])
```

13.6.7. Weitere Funktionen

- 'Teil = Farbe([Farbvorgabe], [Teil])`
  Legt die Anzeigefarbe des Teils fest. Beachten Sie, dass im Gegensatz zu den anderen Funktionen die Definition des Teils in diesem Fall an zweiter Stelle steht.
  Die Farbvorgabe besteht aus den drei RGB-Werten und einer Deckkraft. Zum Beispiel [1,0,0,0.5] für ein Rot mit 50% Deckkraft.
- *myhud = Hud()*
  Erstellt eine Heads-up-Anzeige in der Vismach-GUI, um Elemente wie Achsenpositionen anzuzeigen.
- `tooltip = Capture()`
  Man kann sich dies als ein unsichtbares Bauteil vorstellen, das an der Werkzeugspitze angebracht werden muss, um Position und Orientierung des Werkzeugkoordinatensystems zu verfolgen.

Tatsächlich handelt es sich um eine Transformationsmatrix, die während der Bewegung des Modells fortlaufend aktualisiert wird.

- `work = Capture()`
  Entsprechend wie oben, jedoch am Arbeitstisch angebracht, um das Werkstückkoordinatensystem zu verfolgen.
- `main(model, tooltip, work, size=10, hud=0, rotation_vectors=None, lat=0, lon=0)`
  Dies ist der Befehl, der alles möglich macht, die Anzeige aufbaut usw.
  - *model* sollte eine Sammlung sein, die alle Maschinenteile enthält.
  - *tooltip* und *work* müssen durch `Capture()` erzeugt werden. Vismach benötigt diese Informationen, um den Backplot zu zeichnen, bei dem im Wesentlichen die Werkzeugspitzenposition im Werkstückkoordinatensystem dargestellt wird.
    Ein Beispiel dafür, wie die Werkzeugspitze mit einem Werkzeug und dieses mit dem Modell verbunden wird, findet sich in `scaragui.py`.
  - Entweder *rotation_vectors* oder *latitude/longitude* können verwendet werden, um den ursprünglichen Blickwinkel zu bestimmen, und es ist ratsam, dies zu tun, da der standardmäßige anfängliche Blickwinkel eher wenig hilfreich ist, wenn man direkt über dem Kopf steht.
  - *size* legt die Ausdehnung des in der Ausgangsansicht dargestellten Volumens fest.
  - *hud* bezieht sich auf eine Head-up-Anzeige der Achsenpositionen.

13.6.8. Grundstruktur eines Vismach-Skripts.

```
#imports
from vismach import *
import hal
#create the HAL component and pins
comp = hal.component("compname")
comp.newpin("pin_name", hal.Type.REAL, hal.Dir.IN)
...
#create the floor, tool and work
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()
...
#Build and assemble the model
part1 = Collection([Box(-6,-3,94,6,3,100)])
part1 = Color([1,1,1,1],[part1])
part1 = HalRotate([part1],comp,"pin_name",360,0,0,1)
part1 = Translate([dogs],-1,49,0)
...
#create a top-level model
model = Collection([base, saddle, head, carousel])
#Start the visualization
main(model, tooltip, work, 100, lat=-75, lon=215)
```

Glossar, Copyright & Geschichte

Chapter 14. Umschlagseite

Dieses Handbuch ist noch in Arbeit. Wenn Sie beim Schreiben, Redigieren oder bei der grafischen Aufbereitung helfen können, wenden Sie sich bitte an ein Mitglied des Redaktionsteams oder schreiben Sie eine E-Mail (bevorzugt auf Englisch, aber nicht zwingend, es findet sich jemand) an emc-users@lists.sourceforge.net.

Copyright © 2000-2025 LinuxCNC.org

Es wird die Erlaubnis erteilt, dieses Dokument unter den Bedingungen der GNU Free Documentation License, Version 1.1 oder einer späteren Version, die von der Free Software Foundation veröffentlicht wurde, zu kopieren, zu verbreiten und/oder zu verändern; ohne unveränderliche Abschnitte, ohne Texte auf der Vorderseite und ohne Texte auf der Rückseite des Umschlags. Eine Kopie der Lizenz ist in dem Abschnitt "GNU Free Documentation License" enthalten.

Wenn Sie die Lizenz nicht finden, können Sie eine Kopie bei uns bestellen:

```
Free Software Foundation, Inc.
51 Franklin Street
Fifth Floor
Boston, MA 02110-1301 USA.
```

(Maßgeblich ist die englische Sprachfassung, deswegen wurde sie hier nicht übersetzt, im Fall von Verständnisproblemen siehe zur Anregung http://www.gnu.de/documents/gpl-3.0.de.html und lassen Sie sich beraten)

LINUX® ist das eingetragene Warenzeichen von Linus Torvalds in den USA und anderen Ländern. Die eingetragene Marke Linux® wird im Rahmen einer Unterlizenz von LMI, dem exklusiven Lizenznehmer von Linus Torvalds, dem Eigentümer der Marke auf weltweiter Basis, verwendet.

Das LinuxCNC-Projekt ist nicht mit Debian® verbunden. Debian_ ist ein eingetragenes Warenzeichen im Besitz von Software in the Public Interest, Inc.

Das LinuxCNC-Projekt ist nicht mit UBUNTU® verbunden. UBUNTU ist eine eingetragene Marke im Besitz von Canonical Limited.

Chapter 15. Glossar

Eine Auflistung von Begriffen und deren Bedeutung. Einige Begriffe haben eine allgemeine Bedeutung und mehrere zusätzliche Bedeutungen für Benutzer, Installateure und Entwickler.

Acme-Schraube

Eine Art von Gewindespindel, die ein Acme-Gewinde hat. Acme-Gewinde haben eine etwas geringere Reibung und einen geringeren Verschleiß als einfache Dreiecksgewinde, aber Kugelgewindetriebe sind noch günstiger. Die meisten manuellen Werkzeugmaschinen verwenden Trapezgewindespindeln.

Achse

Eines der computergesteuerten beweglichen Teile der Maschine. Bei einer typischen Vertikalfräse ist der Tisch die X-Achse, der Schlitten die Y-Achse und die Pinole oder das Knie die Z-Achse. Winkelachsen wie Drehtische werden als A, B und C bezeichnet. Zusätzliche lineare Achsen in Bezug auf das Werkzeug heißen U, V und W.

AXIS(GUI)

Eine der grafischen Benutzeroberflächen, die den Benutzern von LinuxCNC zur Verfügung stehen. Es verfügt über die moderne Verwendung von Menüs und Maustasten während der Ausführung und versteckt einige der mehr traditionellen LinuxCNC Kontrollen. Es ist die einzige Open-Source-Schnittstelle, die den gesamten Werkzeugpfad zeigt, sobald eine Datei geöffnet wird.

GMOCCAPY (GUI)

Eine grafische Benutzeroberfläche, die den Benutzern von LinuxCNC zur Verfügung steht. Es bietet die Verwendung und das Gefühl einer industriellen Steuerung und kann mit Touchscreen, Maus und Tastatur verwendet werden. Es unterstützt eingebettete Tabs und von HAL ausgelöste Benutzer-Nachrichten, bietet es eine Menge HAL beens mit Hardware gesteuert werden. GMOCCAPY ist in hohem Maße anpassbar.

Umkehrspiel

Das Spiel oder der Bewegungsverlust, engl. backlash, der bei einer Richtungsumkehr in einer Leitspindel oder einem anderen mechanischen Antriebssystem auftritt. Es kann durch lockere Muttern an Leitspindeln, Schlupf in Riemen, Kabeldurchhang, "Aufwickeln" in Drehkupplungen und anderen Stellen entstehen, an denen das mechanische System nicht "dicht" ist. Spiel führt zu ungenauen Bewegungen, oder im Falle von Bewegungen, die durch äußere Kräfte verursacht werden (z. B. Schneidewerkzeug, das am Werkstück zieht), können Schneidewerkzeuge brechen. Dies kann passieren, weil die Spanbelastung des Fräsers plötzlich ansteigt, wenn das Werkstück durch das Schneidwerkzeug über die Spielstrecke gezogen wird.

Umkehrspiel-Kompensation

Jede Technik, mit der versucht wird, die Auswirkungen des Spiels zu verringern, ohne es tatsächlich aus dem mechanischen System zu entfernen. Dies geschieht in der Regel durch Software in der Steuerung. Auf diese Weise kann die endgültige Ruhestellung des Teils während der Bewegung korrigiert werden, aber Probleme im Zusammenhang mit Richtungsänderungen während der Bewegung (z. B. Kreisinterpolation) und Bewegungen, die durch äußere Kräfte (z. B. Ziehen des Schneidwerkzeugs am Werkstück) verursacht werden, sind nicht gelöst.

Kugelumlaufspindel

Eine Art von Gewindespindel, engl. ball screw, bei der zur Verringerung der Reibung kleine gehärtete Stahlkugeln zwischen der Mutter und der Spindel eingesetzt werden. Kugelgewindetriebe haben eine sehr geringe Reibung und ein geringes Spiel, sind aber in der Regel recht teuer.

Kugelmutter

Eine spezielle Mutter, engl. ball nut, die für die Verwendung mit einer Kugelumlaufspindel bestimmt ist. Sie enthält einen internen Durchgang, um die Kugeln von einem Ende der Spindel zum anderen zurückzubefördern.

CNC

Numerische Computersteuerung (Kurzform für engl. Computational Numerical Control). Allgemeiner Begriff, der sich auf die Computersteuerung von Maschinen bezieht. Statt dass ein menschlicher Bediener Kurbeln dreht, um ein Schneidwerkzeug zu bewegen, verwendet CNC einen Computer und Motoren, um das Werkzeug auf der Grundlage eines Teil-Beschreibung zu bewegen.

Halcompile

Ein Werkzeug, das zum Erstellen, Kompilieren und Installieren von LinuxCNC HAL-Komponenten verwendet wird.

Konfiguration(n)

Ein Verzeichnis, das eine Reihe von Konfigurationsdateien enthält. Benutzerdefinierte Konfigurationen sind in der Regel in den Benutzer home/linuxcnc/configs Verzeichnis gespeichert. Diese Dateien enthalten LinuxCNC's traditionelle INI-Datei und HAL-Dateien. Eine Konfiguration kann auch mehrere allgemeine Dateien enthalten, die Werkzeuge, Parameter und NML-Verbindungen beschreiben.

Konfiguration(v)

Die Aufgabe, LinuxCNC so einzustellen, dass es mit der Hardware einer Werkzeugmaschine übereinstimmt.

Koordinatenmessmaschine

Mit einer Koordinatenmessmaschine (engl. coordinate measuring machine) lassen sich viele genaue Messungen an Teilen vornehmen. Diese Maschinen können verwendet werden, um CAD-Daten für Teile zu erstellen, für die keine Zeichnungen vorhanden sind, wenn ein handgefertigter Prototyp für den Formenbau digitalisiert werden muss, oder um die Genauigkeit von maschinell bearbeiteten oder gegossenen Teilen zu überprüfen.

Anzeigeeinheiten

Die linearen und winkligen Einheiten, die für die Anzeige auf dem Bildschirm verwendet werden.

DRO

Eine digitale Positionsanzeige (Abkürzung von engl. Digital Read Out) ist ein System von Positionsmessgeräten, die an den Schlitten einer Werkzeugmaschine angebracht und mit einer numerischen Anzeige verbunden sind, um die aktuelle Position des Werkzeugs im Verhältnis zu einer Referenzposition anzuzeigen. DROs sind bei handgeführten Werkzeugmaschinen sehr beliebt, da sie die tatsächliche Werkzeugposition spielfrei messen, selbst wenn die Maschine sehr lockere Acme-

Schrauben hat. Einige DROs verwenden lineare Quadratur-Drehgeber, um Positionsinformationen von der Maschine zu erhalten, und einige verwenden Methoden, die einem Resolver ähneln, der immer wieder umläuft.

EDM

Die Funkenerosion (engl. EDM) ist ein Verfahren zum Abtragen von Metall in harten oder schwer zu bearbeitenden oder zähen Metallen oder in Fällen, in denen rotierende Werkzeuge nicht in der Lage wären, die gewünschte Form auf kostengünstige Weise herzustellen. Ein hervorragendes Beispiel sind rechteckige Stanzformen, bei denen scharfe Innenecken gewünscht sind. Bei Fräsvorgängen können mit Werkzeugen mit begrenztem Durchmesser keine scharfen Innenecken erzeugt werden. Eine *Drahterodiermaschine* kann Innenecken mit einem Radius herstellen, der nur geringfügig größer als der Radius des Drahtes' ist. Eine Senkerodiermaschine kann Innenecken mit einem Radius herstellen, der nur geringfügig größer ist als der Radius an der Ecke der Senkelektrode.

EMC

Der Enhanced Machine Controller (ein Eigenname, wörtlich übersetzt "verbesserte Maschinensteuerung"). Ursprünglich ein NIST-Projekt. Umbenannt in LinuxCNC im Jahr 2012.

EMCIO

Das Modul innerhalb von LinuxCNC, die allgemeine E/A (engl. I/O) handhabt, die nichts mit der eigentlichen Bewegung der Achsen zu tun hat.

EMCMOT

Das Modul innerhalb von LinuxCNC, das die eigentliche Bewegung des Schneidwerkzeugs steuert. Es läuft als Echtzeitprogramm und steuert direkt die Motoren.

Encoder

Ein Gerät zur Messung der Position. Normalerweise ein mechanisch-optisches Gerät, das ein Quadratursignal ausgibt. Das Signal kann durch spezielle Hardware gezählt werden, oder direkt durch den parallelen Port mit LinuxCNC.

Vorschub

Relativ langsame, kontrollierte Bewegung des Werkzeugs bei der Durchführung eines Schnitts.

Vorschubgeschwindigkeit

(engl. feed rate) Die Geschwindigkeit, mit der eine Schnittbewegung erfolgt. Im Auto- oder MDI Modus wird die Vorschubgeschwindigkeit mit einem F-Wort bestimmt. F10 würde zehn Maschineneinheiten pro Minute bedeuten.

Rückmeldung

(engl. feedback) Eine Methode (z.B. Quadratur-Encoder-Signale), durch die LinuxCNC Informationen über die Position von Motoren erhält.

Vorschubgeschwindigkeit-Anpassung (engl. override)

Eine manuelle, vom Bediener gesteuerte Änderung des Vorschubs der Geschwindigkeit, mit der sich das Werkzeug beim Schneiden bewegt. Wird oft verwendet, um dem Bediener die Möglichkeit zu geben, stumpfe Werkzeuge oder andere Dinge, die eine Anpassung der Vorschubgeschwindigkeit

erfordern, zu korrigieren.

Gleitkommazahl

A number that has a decimal point. (12.300) In HAL it is known as real.

G-Code

Ein generalisierter Begriff für die gebräuchlichste Programmiersprache zur Beschreibung von Werkstücken. Es gibt mehrere Dialekte von G-Code, LinuxCNC verwendet RS274/NGC.

GUI

Grafische Benutzeroberfläche (engl. Graphical User Interface).

Allgemeines

Eine Art von Schnittstelle zur Kommunikation zwischen einem Computer und einem Menschen (in den meisten Fällen) über die Manipulation von Symbolen und anderen Elementen (Widgets) auf einem Computerbildschirm.

LinuxCNC

Eine Anwendung, die dem Maschinenbediener einen grafischen Bildschirm präsentiert zur Bedienung der Maschine und des Steuerungsprogramms.

HAL

Hardware-Abstraktionsschicht, engl. Abkürzung für *hardware abstraction layer*. Auf der höchsten Ebene ist es einfach eine Möglichkeit, eine Reihe von Bausteinen zu laden und miteinander zu verbinden, um ein komplexes System zusammenzustellen. Viele der Bausteine sind Treiber für Hardwaregeräte. HAL kann jedoch mehr als nur Hardwaretreiber konfigurieren.

Pos1

Eine bestimmte Position im Arbeitsbereich der Maschine, engl. home, die verwendet wird, um sicherzustellen, dass der Computer und die tatsächliche Maschine mit der Werkzeugposition übereinstimmen.

INI-Datei

Eine Textdatei mit dem überwiegenden Anteil an Informationen zur Anpassung (Konfiguration) von LinuxCNC an eine bestimmte Maschine.

Instanz

Man kann eine Instanz einer Klasse oder eines bestimmten Objekts haben. Die Instanz ist das eigentliche Objekt, das zur Laufzeit erzeugt wird. Im Programmierer-Jargon ist das Objekt "Lassie" eine Instanz der Klasse "Dog".

Gelenk-Koordinaten

Diese geben die Winkel zwischen den einzelnen Gelenken der Maschine an. Siehe auch Kinematik

Jog (manuelle Bewegung)

Manuelles Bewegen einer Achse einer Maschine. Beim Joggen wird die Achse entweder bei jedem Tastendruck um einen festen Betrag bewegt oder mit einer konstanten Geschwindigkeit, solange Sie

die Taste gedrückt halten. Im manuellen Modus kann die Jog-Geschwindigkeit über die grafische Oberfläche eingestellt werden.

Kernel-Space

Code, der innerhalb des Kernels läuft, im Gegensatz zu Code, der im Userspace läuft. Einige Echtzeitsysteme (wie RTAI) führen Echtzeitcode im Kernel und Nicht-Echtzeitcode im Userspace aus, während andere Echtzeitsysteme (wie Preempt-RT) sowohl Echtzeit- als auch Nicht-Echtzeitcode im Userspace ausführen.

Kinematik

Die Positionsbeziehung zwischen Weltkoordinaten und Gelenkkoordinaten einer Maschine. Es gibt zwei Arten von Kinematik. Die Vorwärtskinematik wird verwendet, um Weltkoordinaten aus Gelenkkoordinaten zu berechnen. Die inverse Kinematik wird für genau den gegenteiligen Zweck verwendet. Beachten Sie, dass die Kinematik die Kräfte, Momente usw. an der Maschine nicht berücksichtigt. Sie dient nur der Positionierung.

Leitspindel

Eine Spindel, die von einem Motor gedreht wird, um einen Tisch oder einen anderen Teil einer Maschine zu bewegen. Gewindespindeln sind in der Regel entweder Kugelgewindespindeln oder Trapezgewindespindeln, obwohl auch herkömmliche dreieckige Gewindespindeln verwendet werden können, wenn Genauigkeit und lange Lebensdauer weniger wichtig sind als niedrige Kosten.

Maschineneinheiten

Die für die Maschinenkonfiguration verwendeten Längen- und Winkeleinheiten. Diese Einheiten werden in der INI-Datei angegeben und verwendet. HAL-Pins und -Parameter werden im Allgemeinen ebenfalls in Maschineneinheiten angegeben.

MDI

Manuelle Dateneingabe (engl. Abkürzung für Manual Data Input). Dies ist eine Betriebsart, bei der das Steuergerät einzelne Zeilen des G-Codes ausführt, wie sie vom Bediener eingegeben werden.

NIST

Das US Institut "Nationales Institut für Normung und Technologie" (engl. Abkurzung für National Institute of Standards and Technology). Eine Einrichtung des Handelsministeriums der Vereinigten Staaten.

NML

Die Neutral Message Language bietet einen Mechanismus für die Handhabung mehrerer Nachrichtentypen im selben Puffer sowie eine Vereinfachung der Schnittstelle für die Kodierung und Dekodierung von Puffern im neutralen Format und des Konfigurationsmechanismus.

Versätze

Ein beliebiger Betrag, der zum Wert von etwas hinzugefügt wird, um ihn mit einem gewünschten Wert gleichzusetzen. Zum Beispiel werden G-Code-Programme oft um einen geeigneten Punkt herum geschrieben, wie X0, Y0. Vorrichtungsoffsets können verwendet werden, um den tatsächlichen Ausführungspunkt dieses G-Code-Programms so zu verschieben, dass er mit der tatsächlichen Position des Schraubstocks und der Backen übereinstimmt. Werkzeugkorrekturen können verwendet

werden, um die unkorrigierte Länge eines Werkzeugs so zu verschieben, dass sie der tatsächlichen Länge des Werkzeugs entspricht.

Werkstück Programm

Beschreibung eines Werkstücks in einer Sprache, welche die Steuerung verstehen kann. Für LinuxCNC ist die Sprache RS-274/NGC, allgemein als G-Code bekannt.

Programm-Einheiten

Die in einem Werkstück-Programm verwendeten Längen- und Winkeleinheiten. Die linearen Programmeinheiten müssen nicht mit den linearen Maschineneinheiten übereinstimmen. Siehe G20 und G21 für weitere Informationen. Die Winkeleinheiten des Programms werden immer in Grad gemessen.

Python

Allzweck-, sehr High-Level-Programmiersprache. Wird in LinuxCNC verwendet für die Axis GUI, das StepConf Konfigurationswerkzeug, und mehrere G-Code-Programmierung Skripte.

Schnell

Schnelle, möglicherweise unpräzise Bewegung des Werkzeugs, die in der Regel für den Wechsel zwischen den Schnitten verwendet wird. Wenn das Werkzeug beim Eilgang auf das Werkstück oder die Vorrichtung trifft, ist das wahrscheinlich schlecht!

Schnellauf-Geschwindigkeit

Die Geschwindigkeit, mit der eine Eilgangbewegung erfolgt. Im Auto- oder MDI-Modus ist der Eilgang normalerweise die Höchstgeschwindigkeit der Maschine. Es ist oft wünschenswert, die Eilgeschwindigkeit zu begrenzen, wenn ein G-Code-Programm zum ersten Mal getestet wird.

Echtzeit

Software, die sehr strenge Zeitvorgaben einhalten soll. Um diese Anforderungen zu erfüllen, ist es unter Linux notwendig, einen Echtzeit-Kernel wie RTAI oder Preempt-RT zu installieren und die LinuxCNC-Software so zu erstellen, dass sie in der speziellen Echtzeitumgebung läuft. Echtzeitsoftware kann im Kernel oder im Userspace laufen, abhängig von den Möglichkeiten, die das System bietet.

RTAI

Real Time Application Interface, siehe https://www.rtai.org/ mit Echtzeit-Erweiterungen für Linux, die LinuxCNC verwenden kann, um Echtzeit-Leistung zu erreichen.

RTLINUX

Siehe https://en.wikipedia.org/wiki/RTLinux, eine ältere Echtzeit-Erweiterung für Linux, die von LinuxCNC verwendet wurde, um Echtzeitleistung zu erreichen. Veraltet, ersetzt durch RTAI.

RTAPI

Eine portable Schnittstelle zu Echtzeitbetriebssystemen einschließlich RTAI und POSIX pthreads mit Echtzeit-Erweiterungen.

RS-274/NGC

Der formale Name für die Sprache, die von LinuxCNC-Werkstück-Programmen verwendet wird.

Servomotor

Im Allgemeinen ein Motor, der über Fehlererkennung die Korrektur der Position eines Stellglieds vornimmt. Auch ein Motor, der speziell für eine verbesserte Leistung in solchen Anwendungen ausgelegt ist.

Servo Loop

Engl. für "Schleife", hier ein Regelkreis zur Steuerung der Position oder der Geschwindigkeit eines Motors, der mit einer Rückkopplungseinrichtung ausgestattet ist.

Ganze Zahl mit Vorzeichen

A whole number that can have a positive or negative sign. In HAL it is a sint.

Spindel

Der Teil einer Werkzeugmaschine, der sich dreht, um den Schnitt auszuführen. Bei einer Fräs- oder Bohrmaschine hält die Spindel das Schneidwerkzeug. Bei einer Drehmaschine hält die Spindel das Werkstück.

Spindeldrehzahl-Anpassung

Eine manuelle, vom Bediener gesteuerte Änderung der Geschwindigkeit, mit der sich das Werkzeug während des Schneidens dreht. Oft verwendet, um dem Bediener zu ermöglichen, für Ratter verursacht durch die cutter’s Zähne anzupassen. Spindeldrehzahl Override setzt voraus, dass die LinuxCNC-Software dafür konfiguriert wurde, die Spindeldrehzahl zu steuern.

StepConf

Ein LinuxCNC Konfigurations-Assistent. Es ist in der Lage, viele Schritt-und-Richtung Bewegung Befehl basierte Maschinen zu behandeln. Er schreibt eine vollständige Konfiguration, nachdem der Benutzer ein paar Fragen über den Computer und die LinuxCNC-ausführenden Maschine beantwortet hat.

Schrittmotor

Eine Art von Motor, der sich in festen Schritten dreht. Durch Zählen der Schritte lässt sich feststellen, wie weit sich der Motor gedreht hat. Wenn die Last die Drehmomentkapazität des Motors übersteigt, überspringt er einen oder mehrere Schritte, was zu Positionsfehlern führt.

TASK (engl. für Aufgabe, auch Name des entsprechenden LinuxCNC Moduls)

Das Modul innerhalb von LinuxCNC, das die gesamte Ausführung koordiniert und das Teileprogramm interpretiert.

Tcl/Tk

Eine Skriptsprache und ein grafisches Widget-Toolkit, mit dem mehrere der LinuxCNC-GUIs und Auswahl-Assistenten geschrieben wurden.

Traverse Bewegung

Eine Bewegung in gerader Linie vom Startpunkt zum Endpunkt.

Einheiten

Siehe "Maschineneinheiten", "Anzeigeeinheiten", oder "Programmeinheiten".

Ganzzahl ohne Vorzeichen

A whole number that has no sign. In HAL it is a uint.

Weltkoordinaten

Dies ist der absolute Bezugsrahmen. Es gibt die Koordinaten in Bezug auf einen festen Bezugsrahmen an, der an einem Punkt (im Allgemeinen der Basis) der Werkzeugmaschine befestigt ist.

Chapter 16. Copyright

16.1. Juristischer Abschnitt

Die Übersetzungen dieser Datei im Quellbaum sind nicht rechtsverbindlich.

16.1.1. Copyright-Bedingungen

Copyright (c) 2000-2022 LinuxCNC.org

Es wird die Erlaubnis erteilt, dieses Dokument unter den Bedingungen der GNU Free Documentation License, Version 1.1 oder einer späteren Version, die von der Free Software Foundation veröffentlicht wurde, zu kopieren, zu verbreiten und/oder zu verändern; ohne unveränderliche Abschnitte, ohne Texte auf der Vorderseite und ohne Texte auf der Rückseite des Umschlags. Eine Kopie der Lizenz ist in dem Abschnitt "GNU Free Documentation License" enthalten.

16.1.2. GNU Free Documentation License

GNU Free Documentation License Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Es ist jedermann gestattet, wortwörtliche Kopien dieses Lizenzdokuments zu kopieren und zu verbreiten, aber es ist nicht erlaubt, es zu verändern.

0. PREAMBEL

Der Zweck dieser Lizenz ist es, ein Handbuch, ein Lehrbuch oder ein anderes schriftliches Dokument "frei" im Sinne von Freiheit zu machen: jedem die effektive Freiheit zu sichern, es zu kopieren und weiterzugeben, mit oder ohne Modifikation, entweder kommerziell oder nicht-kommerziell. In zweiter Linie bewahrt diese Lizenz dem Autor und dem Herausgeber eine Möglichkeit, Anerkennung für ihre Arbeit zu erhalten, während sie nicht für die von anderen vorgenommenen Änderungen verantwortlich gemacht werden.

Diese Lizenz ist eine Art "Copyleft", was bedeutet, dass abgeleitete Werke des Dokuments selbst im gleichen Sinne frei sein müssen. Sie ergänzt die GNU General Public License, die eine Copyleft-Lizenz für freie Software ist.

Wir haben diese Lizenz entworfen, um sie für Handbücher für freie Software zu verwenden, weil freie Software freie Dokumentation braucht: ein freies Programm sollte mit Handbüchern geliefert werden, welche die gleichen Freiheiten bieten wie die Software. Aber diese Lizenz ist nicht auf Software-Handbücher beschränkt; sie kann für jedes textliche Werk verwendet werden, unabhängig vom Thema oder ob es als gedrucktes Buch veröffentlicht wird. Wir empfehlen diese Lizenz in erster Linie für Werke, deren Zweck die Anleitung oder das Nachschlagen ist.

1. ANWENDBARKEIT UND DEFINITIONEN

Diese Lizenz gilt für jedes Handbuch oder andere Werk, das einen Hinweis des Urheberrechtsinhabers enthält, der besagt, dass es unter den Bedingungen dieser Lizenz verbreitet werden darf. Das

"Dokument", unten, bezieht sich auf ein solches Handbuch oder Werk. Jedes Mitglied der Öffentlichkeit ist ein Lizenznehmer und wird als "Sie" angesprochen.

Eine "modifizierte Version" des Dokuments ist jedes Werk, welches das Dokument oder einen Teil davon enthält, entweder wortwörtlich kopiert oder mit Änderungen und/oder in eine andere Sprache übersetzt.

Ein "sekundärer Abschnitt" ist ein benannter Anhang oder ein vorderer Abschnitt des Dokuments, der sich ausschließlich mit der Beziehung der Herausgeber oder Autoren des Dokuments zum Gesamtthema des Dokuments (oder zu verwandten Themen) befasst und nichts enthält, was direkt in dieses Gesamtthema fallen könnte. (Wenn das Dokument zum Beispiel teilweise ein Lehrbuch der Mathematik ist, darf ein sekundärer Abschnitt keine Mathematik erklären). Die Beziehung könnte eine Frage des historischen Zusammenhangs mit dem Thema oder mit verwandten Themen oder der rechtlichen, kommerziellen, philosophischen, ethischen oder politischen Position dazu sein.

Die unveränderlichen Abschnitte" sind bestimmte sekundäre Abschnitte, deren Titel in der Mitteilung, die besagt, dass das Dokument unter dieser Lizenz freigegeben ist, als die der unveränderlichen Abschnitte bezeichnet werden.

Die "Covertexte" sind bestimmte kurze Textpassagen, die als Front-Cover-Texte oder Back-Cover-Texte in dem Hinweis aufgeführt sind, der besagt, dass das Dokument unter dieser Lizenz freigegeben ist.

Eine "transparente" Kopie des Dokuments ist eine maschinenlesbare Kopie, die in einem Format dargestellt wird, dessen Spezifikation der Allgemeinheit zur Verfügung steht, dessen Inhalt direkt und unkompliziert mit allgemeinen Texteditoren oder (für Bilder, die aus Pixeln bestehen) mit allgemeinen Malprogrammen oder (für Zeichnungen) mit einem weit verbreiteten Zeichnungseditor betrachtet und bearbeitet werden kann, und die für die Eingabe in Textformatierer oder für die automatische Übersetzung in eine Vielzahl von Formaten geeignet ist, die für die Eingabe in Textformatierer geeignet sind. Eine Kopie, die in einem ansonsten transparenten Dateiformat erstellt wurde, dessen Markup so gestaltet wurde, dass eine nachträgliche Änderung durch Leser vereitelt oder erschwert wird, ist nicht transparent. Eine Kopie, die nicht "Transparent" ist, wird als "Opak" bezeichnet.

Geeignete Formate für transparente Kopien sind z. B. ASCII ohne Markup, Texinfo-Eingabeformat, LaTeX-Eingabeformat, SGML oder XML mit einer öffentlich zugänglichen DTD und standardkonformes einfaches HTML, das für die Bearbeitung durch den Menschen ausgelegt ist. Zu den undurchsichtigen Formaten gehören PostScript, PDF, proprietäre Formate, die nur von proprietären Textverarbeitungsprogrammen gelesen und bearbeitet werden können, SGML oder XML, für die eine DTD und/oder die Verarbeitungswerkzeuge nicht allgemein verfügbar sind, und das maschinell erzeugte HTML, das von einigen Textverarbeitungsprogrammen nur zu Ausgabezwecken erzeugt wird.

Die "Titelseite" bedeutet bei einem gedruckten Buch die Titelseite selbst sowie die Folgeseiten, die benötigt werden, um das Material, das nach dieser Lizenz auf der Titelseite erscheinen soll, lesbar zu halten. Für Werke in Formaten, die kein Titelblatt als solches haben, bedeutet "Titelblatt" den Text in der Nähe des auffälligsten Erscheinens des Werktitels, der dem Beginn des Textes vorausgeht.

2. WORTWÖRTLICHES KOPIEREN

Sie dürfen das Dokument in jedem beliebigen Medium kopieren und verbreiten, sei es kommerziell oder nicht kommerziell, vorausgesetzt, dass diese Lizenz, die Urheberrechtsvermerke und der

Lizenzvermerk, der besagt, dass diese Lizenz für das Dokument gilt, in allen Kopien wiedergegeben werden, und dass Sie keine weiteren Bedingungen zu denen dieser Lizenz hinzufügen. Sie dürfen keine technischen Maßnahmen anwenden, um das Lesen oder weitere Kopieren der von Ihnen erstellten oder verbreiteten Kopien zu behindern oder zu kontrollieren. Sie dürfen jedoch eine Vergütung im Austausch für Kopien annehmen. Wenn Sie eine ausreichend große Anzahl von Kopien verbreiten, müssen Sie auch die Bedingungen in Abschnitt 3 einhalten.

Sie können auch Kopien unter den oben genannten Bedingungen ausleihen und öffentlich ausstellen.

3. MASSENHAFTES KOPIEREN

Wenn Sie mehr als 100 gedruckte Exemplare des Dokuments veröffentlichen und der Lizenzhinweis des Dokuments Umschlagtexte verlangt, müssen Sie die Exemplare in Umschläge einlegen, die deutlich und lesbar alle diese Umschlagtexte enthalten: Vorderseitentexte auf dem vorderen Umschlag und Rückseitentexte auf dem hinteren Umschlag. Auf beiden Umschlägen müssen Sie außerdem deutlich und leserlich als Verleger dieser Exemplare ausgewiesen sein. Der vordere Umschlag muss den vollständigen Titel enthalten, wobei alle Wörter des Titels gleichmäßig hervorgehoben und sichtbar sein müssen. Sie können die Umschläge zusätzlich mit anderem Material versehen. Kopien mit Änderungen, die sich auf die Umschläge beschränken, können als wortgetreue Kopien behandelt werden, solange der Titel des Dokuments erhalten bleibt und diese Bedingungen erfüllt sind.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. VERÄNDERUNGEN

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

1. Verwenden Sie auf der Titelseite (und auf den Umschlägen, falls vorhanden) einen Titel, der sich von dem des Dokuments und von denen früherer Versionen unterscheidet (die, falls es welche gab, im Abschnitt "Historie" des Dokuments aufgeführt sein sollten). Sie können denselben Titel wie eine frühere Version verwenden, wenn der ursprüngliche Herausgeber dieser Version seine Zustimmung

gibt. B. Führen Sie auf der Titelseite als Autoren eine oder mehrere Personen oder Organisationen auf, die für die Urheberschaft der Änderungen in der geänderten Version verantwortlich sind, zusammen mit mindestens fünf der Hauptautoren des Dokuments (alle seine Hauptautoren, wenn es weniger als fünf hat). C. Geben Sie auf der Titelseite den Namen des Herausgebers der geänderten Version als Herausgeber an. D. Behalten Sie alle Urheberrechtsvermerke des Dokuments bei. E. Fügen Sie einen angemessenen Urheberrechtsvermerk für Ihre Änderungen neben den anderen Urheberrechtsvermerken ein. F. Fügen Sie unmittelbar nach den Urheberrechtsvermerken einen Lizenzhinweis ein, welcher der Öffentlichkeit die Erlaubnis gibt, die modifizierte Version unter den Bedingungen dieser Lizenz zu benutzen, und zwar in der Form, die im Anhang unten gezeigt wird. G. Behalten Sie in diesem Lizenzhinweis die vollständigen Listen der unveränderlichen Abschnitte und der erforderlichen Umschlagtexte bei, die im Lizenzhinweis des Dokuments angegeben sind. H. Fügen Sie eine unveränderte Kopie dieser Lizenz bei. I. Behalten Sie den Abschnitt mit dem Titel Geschichte" und seinen Titel bei und fügen Sie ihm einen Punkt hinzu, der mindestens den Titel, das Jahr, die neuen Autoren und den Herausgeber der modifizierten Version angibt, wie auf der Titelseite angegeben. Wenn es keinen Abschnitt mit dem Titel "Geschichte" in dem Dokument gibt, erstellen Sie einen, der den Titel, das Jahr, die Autoren und den Herausgeber des Dokuments angibt, wie auf der Titelseite angegeben, und fügen Sie dann einen Punkt hinzu, der die geänderte Version beschreibt, wie im vorherigen Satz angegeben. J. Bewahren Sie den im Dokument angegebenen Netzwerkstandort, falls vorhanden, für den öffentlichen Zugang zu einer transparenten Kopie des Dokuments auf, und ebenso die im Dokument angegebenen Netzwerkstandorte für frühere Versionen, auf denen es basierte. Diese können im Abschnitt "Historie" abgelegt werden. Sie können eine Netzwerkadresse für ein Werk weglassen, das mindestens vier Jahre vor dem Dokument selbst veröffentlicht wurde, oder wenn der ursprüngliche Herausgeber der Version, auf die es sich bezieht, die Erlaubnis gibt. K. In jedem Abschnitt, der mit "Danksagungen" oder "Widmungen" betitelt ist, bewahren Sie den Titel des Abschnitts, und bewahren Sie in dem Abschnitt den gesamten Inhalt und Ton der Danksagungen und/oder Widmungen, die darin enthalten sind. L. Bewahren Sie alle unveränderlichen Abschnitte des Dokuments, unverändert in ihrem Text und in ihren Titeln. Abschnittsnummern oder das Äquivalent werden nicht als Teil der Abschnittstitel betrachtet. M. Streichen Sie jeden Abschnitt mit der Überschrift "Vermerke". Ein solcher Abschnitt darf nicht in die geänderte Fassung aufgenommen werden. N. Vorhandene Abschnitte dürfen nicht in "Vermerke" umbenannt werden oder im Titel mit einem unveränderlichen Abschnitt kollidieren.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. ÜBERSETZUNG

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission

from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINIERUNG

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. ZUKÜNFTIGE ÜBERARBEITUNGEN DIESER LIZENZ

Die Free Software Foundation kann von Zeit zu Zeit neue, überarbeitete Versionen der GNU Free Documentation License veröffentlichen. Solche neuen Versionen werden im Geiste der gegenwärtigen Version ähnlich sein, können sich aber im Detail unterscheiden, um neue Probleme oder Anliegen zu behandeln. Siehe https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: Wie diese Lizenz für Ihre Dokumente anzuwenden ist

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Chapter 17. LinuxCNC Geschichte

17.1. Origin (engl. für Ursprung)

EMC (der Enhanced Machine Controller) wurde von NIST , dem National Institute of Standards and Technology, einer Behörde des Handelsministeriums der Vereinigten Staaten, entwickelt.

Das NIST interessierte sich zunächst für die Entwicklung eines Bewegungssteuerungspakets als Testplattform für Konzepte und Normen. Die frühe Förderung durch General Motors führte zu einer Anpassung der noch jungen Version von EMC unter Verwendung intelligenter PMAC-Steuerungskarten, die unter einer "Echtzeit"version von Windows NT liefen und eine große Fräsmaschine steuerten.

Wie bei allen *Arbeitsprodukten* von Mitarbeitern der US-Bundesregierung vorgeschrieben, müssen die resultierende Software und der Bericht darüber öffentlich zugänglich sein, und ein Bericht darüber wurde ordnungsgemäß veröffentlicht, auch im Internet. Dort entdeckte Matt Shaver EMC. Er setzte sich mit dem NIST in Verbindung und diskutierte mit Fred Proctor über die Anpassung des Codes für die Steuerung preiswerterer Hardware, die für die Aufrüstung und den Ersatz veralteter oder schlichtweg toter CNC-Steuerungen verwendet werden sollte. Das NIST war fasziniert, denn auch sie wollten etwas Günstigeres. Um eine Zusammenarbeit in die Wege zu leiten, wurde eine formelle Vereinbarung getroffen, die garantierte, dass der resultierende Code und das Design public domain bleiben würden.

Die ersten Überlegungen konzentrierten sich auf den Ersatz des teuren und temperamentvollen Windows-NT-Systems (Echtzeit). Es wurde vorgeschlagen, eine (damals) relativ neue Echtzeit-Erweiterung des Linux-Betriebssystems zu testen. Diese Idee wurde mit Erfolg weiterverfolgt. Als Nächstes stand die Frage der teuren intelligenten Bewegungssteuerungskarten an. Zu diesem Zeitpunkt wurde die Rechenleistung eines PCs als groß genug angesehen, um die Kontrolle über die Bewegungsroutinen direkt zu übernehmen. Eine schnelle Suche nach verfügbarer Hardware führte zur Auswahl einer "Servo-To-Go"-Schnittstellenkarte als erste Plattform, mit der PC die Motoren direkt steuern konnte. Die vorhandene Benutzeroberfläche und der RS274-Interpreter wurden um Software für die Bahnplanung und PID-Regelung ergänzt. Matt setzte diese Version erfolgreich ein, um einige Maschinen mit toten Steuerungen aufzurüsten, und dies wurde das EMC-System, das die Aufmerksamkeit der Außenwelt auf sich zog. Die Erwähnung von EMC in der USENET-Newsgroup rec.crafts.metalworking führte dazu, dass frühe Anwender wie Jon Elson Systeme bauten, um die Vorteile von EMC zu nutzen.

Das NIST richtete eine Mailingliste für Personen ein, die sich für EMC interessierten. Im Laufe der Zeit interessierten sich auch andere außerhalb des NIST für die Verbesserung von EMC. Viele Leute baten um kleine Verbesserungen des Codes oder programmierten sie. Ray Henry wollte die Benutzeroberfläche verfeinern. Da Ray sich nicht traute, den C-Code, in dem die Benutzeroberfläche geschrieben war, zu verändern, wurde eine einfachere Methode gesucht. Fred Proctor vom NIST schlug eine Skriptsprache vor und schrieb einen Code, um die Skriptsprache Tcl/Tk mit der internen NML-Kommunikation von EMC zu verbinden. Mit diesem Tool schrieb Ray dann ein Tcl/Tk-Programm, das zur damals vorherrschenden Benutzeroberfläche für EMC wurde.

Die Perspektive des NIST finden Sie in diesem paper von William Shackleford und Frederick Proctor, das die Geschichte von EMC und den Übergang zu Open Source beschreibt.

Zu dieser Zeit begann das Interesse an EMC erheblich zuzunehmen. Als immer mehr Leute versuchten,

EMC zu installieren, wurde die Schwierigkeit, einen Linux-Kernel mit den Echtzeit-Erweiterungen zu patchen und den EMC-Code zu kompilieren, überdeutlich. Viele Versuche, den Prozess zu dokumentieren und Skripte zu schreiben, wurden unternommen, einige mit mäßigem Erfolg. Immer wieder tauchte das Problem auf, die richtige Version der Patches und Compiler mit der ausgewählten Linux-Version abzugleichen. Paul Corner kam zur Rettung mit der BDI (brain dead install), einer CD, von der ein komplettes funktionierendes System (Linux, Patches und EMC) installiert werden konnte. Der BDI-Ansatz öffnete die Welt der EMC für eine viel größere Benutzergemeinschaft. Als diese Gemeinschaft weiter wuchs, wurden die EMC-Mailingliste und die Code-Archive nach SourceForge verschoben und die LinuxCNC-Website eingerichtet.

Durch die Teilnahme einer größeren Gemeinschaft von Anwendern wurde EMC zu einem wichtigen Interessenschwerpunkt bei den laufenden CNC-Ausstellungen auf der NAMES, und die NAMES wurde zur jährlichen Veranstaltung für EMC. In den ersten Jahren fanden die Treffen nur deshalb statt, weil die interessierten Parteien auf der NAMES waren. Im Jahr 2003 fand das erste angekündigte öffentliche Treffen der EMC-Anwendergemeinschaft statt. Es fand am Montag nach der NAMES in der Lobby der Arena statt, in der auch die NAMES-Messe abgehalten wurde. Die Organisation war locker, aber die Idee einer Hardware-Abstraktionsschicht (HAL) wurde geboren und die Umstrukturierung des Codes zur Vereinfachung der Entwicklung (EMC2) wurde vorgeschlagen.

17.1.1. Namensänderung

Im Frühjahr 2011 wurde der LinuxCNC-Vorstand von einer Anwaltskanzlei in Vertretung der EMC Corporation (www.emc.com) wegen der Verwendung von "EMC" und "EMC2" zur Kennzeichnung der auf linuxcnc.org angebotenen Software kontaktiert. Die EMC Corporation hat verschiedene Marken im Zusammenhang mit EMC und EMC² (EMC mit hochgestellter Zahl zwei) eingetragen.

Nach einer Reihe von Gesprächen mit dem Vertreter der EMC Corporation war das Endergebnis, dass mit dem nächsten großen Release der Software, linuxcnc.org aufhörte, die Identifizierung der Software mit "emc", "EMC", oder diese Zeichenfolgen in Kombination mit nachfolgenden Ziffern zu nutzen. Soweit das LinuxCNC Board of Directors eine Kontrolle hat über die Bezeichnung der auf linuxcnc.org angeboten Software, hat der Vorstand diesem zugestimmt.

Infolgedessen war es notwendig, einen neuen Namen für die Software zu wählen. Von den Optionen, die der Vorstand in Betracht zog, gab es einen Konsens, dass "LinuxCNC" die beste Option ist, da dies seit Jahren der Name unserer Website ist'.

In Vorbereitung auf den neuen Namen haben wir eine Unterlizenz für die Marke LINUX® von der Linux Foundation (www.linuxfoundation.org) erhalten, die unsere Verwendung des Namens LinuxCNC schützt. (LINUX® ist die eingetragene Marke von Linus Torvalds in den USA und anderen Ländern.)

Die Umbenennung umfasste die Website linuxcnc.org, die IRC-Kanäle und die Versionen der Software und der Dokumentation seit Version 2.5.0.

17.1.2. Zusätzliche Informationen

Das NIST hat ein Papier veröffentlicht, in dem die Sprache RS274NGC und das von ihr gesteuerte abstrakte Bearbeitungszentrum sowie eine frühe Implementierung von EMC beschrieben werden. Das Papier ist auch unter https://linuxcnc.org/files/RS274NGCv3.pdf verfügbar.

Das NIST hat auch ein Papier über die Geschichte der EMC und ihren Übergang zu open source veröffentlicht. Das Papier ist auch unter https://linuxcnc.org/files/Use-of-Open-Source-Distribution-for-a-Machine-Tool-Controller.pdf verfügbar